THE LEARNING OF A COMPUTER PROGRAMMING LANGUAGE BY SECOND AND THIRD GRADERS WITH SELF-PACED MATERIALS

by

Nancy Austin Shuller

Dissertation Committee:

Professor Bruce R. Vogeli, Sponsor

Professor Robert P. Taylor

Approved by the Committee on the Degree of Doctor of Education

Date MAY 1 4 1984

Submitted in partial fulfillment of the requirements for the Degree of Doctor of Education in Teachers College, Columbia University

1984

ABSTRACT

THE LEARNING OF A COMPUTER PROGRAMMING LANGUAGE
BY SECOND AND THIRD GRADERS WITH SELF-PACED MATERIALS

Nancy Austin Shuller

This research study examines the effectiveness of self-paced materials to enable students in grades two and three to learn the programming language, Atari PILOT. After twelve hours of in-class and laboratory computer time, the students were compared to determine the extent of self-pacing, kind and amount of assistance needed, programming ability, attitudes towards their work with computers, and the extent to which programming ability was influenced by other factors.

During the field trial period, observations were made and records kept by the classroom teachers and the laboratory instructor as the children worked in their materials and/or on the computer. At the conclusion of that period, all students wrote an original program for evaluation and categorization and were given an attitude survey. A self-selected group of children spent an additional two hours working in PILOT in an afterschool computer program.

The range of materials completed indicated that to a rather high extent self-pacing was achieved. Children relied on assistance more in the beginning of the field trial. As students' confidence in their ability to control the computer, write programs, and work through the materials grew, less assistance was needed. Class differences in

completion of materials were significant.

While all students wrote programs, the range was extensive from a simple copied program to a more sophisticated one using programming structures and showing evidence of planning. The three levels of programs written, among which the sample was almost equally divided, were labeled imitative (low), exploratory (middle), and planned (high). It was shown that programming ability was significantly affected by the amount of learning materials completed, motivation to begin or continue work on the computer, and amount of time on the computer. Individual teachers affected the amount of materials completed as well as programming ability. It was shown that provision for equal computer access, assistance when needed, and the presence of expertise may be critical elements for programming success.

This dissertation is dedicated to the staff and children of The Day School who provided me with their time, insights, questions, support, understanding, and enthusiasm so that this research project might be realized.

N. A. S.

Table of Contents

List of Tables

List of Figures

Chapter I

INTRODUCTION

Need for the Study

Entry into the business, home, or school market by any new technology often produces questions of utility within the other markets, especially when availability comes within the financial means of many. With increased use of microcomputers in business and the home, the potential of the microcomputer to improve — or even revolutionize — the school is currently being explored. As with every other technological innovation to find its way into the classroom, a myriad of educational uses for microcomputers is being suggested. Many discussions center around control. Should the computer control the student or should the student control the computer?

In programming the computer students clearly have control. Computer programming is rapidly becoming an integral part of the high school curriculum. Yet as computer accessibility increases through user-friendly languages and cost-friendly prices, and the intellectual benefits reaped are realized, the question of whether the computer should also be available to elementary and junior high students has been raised. In these situations, where should the control lie?

The fact that computer programming should be an integral part of every child's school experience has been stated implicitly in the National Council of Teachers of Mathematics' Agenda for Action: Recommendations for the 1980s.

> Recommendation 3: Mathematics programs must take full advantage of the power of calculators and computers at all grade levels.[1]

Surely to take full advantage of the power of the computer includes the ability to program the computer. Much is said to justify teaching computer programming in grades K-12 and for many diverse reasons, ranging from providing a computer literate citizenry to enhancing the students' intellectual abilities. Arthur Luehrmann, formerly an Associate Director of the Lawrence Hall of Science, is a noted authority on computers in education. As an advocate of computer programming in the schools, he summarizes his feelings in this way:

> Computing belongs as a regular school subject for the same reason that reading, writing, and mathematics are there. Each one gives the student a basic intellectual tool with wide areas of application. Each one gives the student a distinctive means of thinking about and representing a problem, of writing his or her thoughts down, of studying and criticizing the thoughts of others, and of rethinking and revising ideas, whether they are embodied in a paragraph of English, a set of mathematical equations, or a computer program.[2]

As easily accessible, yet powerful languages become available for the microcomputer, computer programming is becoming an integral part of the curriculum for many elementary school children. Simple programmable toys and/or robots like the Turtle or Big Trak, and languages like Logo, PILOT, and Kidstuff are enabling young children to "teach" computers, as well as providing them with insights into their learning. The computer language, Logo, developed at MIT by Seymour Papert, is

---

[1] National Council of Teachers of Mathematics, Agenda for Action: Recommendations for School Mathematics of the 1980's (Reston: NCTM, 1980), p. 8.

[2] A. Luehrmann. "Computer Literacy - What Should It Be?" The Mathematics Teacher 74 (1981), p. 686.

an attempt to create a vehicle through which students are put in control of their own learning, via the student-controlled movement of a turtle on the screen. As students simulate and discuss turtle movements, and then execute these movements on the computer, they are, according to Papert

> getting used to the idea of heuristic knowledge, they are learning to think of mathematics as rooted in (not opposed to) intuitive body-mathematics and they are using mathematics as a language.[3]

In this paper, the languages Logo, PILOT, Smalltalk, and Kidstuff will be called turtle graphics languages, since turtle graphics is one of their major components and that aspect of the language which makes clear many commands and structures via imagery produced on the screen. PILOT, while being procedurally different from Logo, offers its users many of the same capabilities for exploring their world mathematically through these turtle graphics components.

Curriculum projects have been developed to provide students with an opportunity to learn Logo. Learning materials, of varying degrees of excellence, exist for all turtle graphics languages in the form of manuals or student booklets useful for the older student, adult, or teacher. Due to readibility and/or design, existing materials cannot be used directly or independently by younger students.

There exists, therefore, a need to develop learning materials specifically designed for children in the early elementary grades to provide them with the necessary support to learn these turtle graphics languages. The materials developed need to take into account the

---

[3] S. Papert. "New Cultures from New Technologies" BYTE 5 (1980), p. 113.

perspective that learning is a primary natural function of the mind, and instruction, if it is to exist at all, needs to allow the individuals the right to delve into problems, search for solutions, test theories, and take responsibility for one's own learning. The Papertian idea of a microworld, a computer-based exploration that serves as an entry into a greater wealth of knowledge, should be one aspect of materials for use in learning a turtle-graphics language. While there are no materials of this type for any of these turtle graphics languages, an attempt to fill the gap in one such language will surely provide material, ideas, and/or techniques for use within the others.

Atari PILOT, a user-friendly turtle graphics language, incorporates much of the spirit of Logo. Atari PILOT, available for over a year, has no curriculum materials developed for it. Its manual and Student Pilot do not provide adequate support materials for either students or teachers.

Purpose

The purpose of this study is to prepare self-paced materials designed to be used independently by children of the early elementary grades so as to enable them to learn the Atari PILOT language, and to submit these materials to a field trial by second and third grade students at a private school in New York City.

Investigations

The following questions will be investigated as a part of the

field trial:

1. In regards to learning materials completion, what is the a) achievement level, b) extent of self-pacing achieved, and c) kind and amount of assistance given?

2. After approximately six hours of in-class use and six hours of laboratory sessions with the materials, what kind of programs are children able to write in thirty minutes when given instructions to "write an interesting program for the computer all by yourself"?

3. To what extent are students' attitudes concerning their work with computers affected?

4. To what extent is programming ability influenced by sex, grade level, teacher, students' attitudes about themselves and their computer work, standardized reading and mathematics scores, amount of learning materials completed, or time spent working on the computer?

Procedures

Many procedures were completed prior to the field trial. The literature was reviewed, objectives formulated, and field trial site selected. The learning materials were developed in parts, each part introducing new commands and programming ideas, designed so as to build on information from previous parts. The parental questionnaire was devised and disseminated to determine possible contamination to the study by the use of computers and/or Atari PILOT in the home. The following instruments were developed: a teacher reporting form on which classroom teachers would record information concerning the use of the materials by the students; an individual student achievement graph

to record the progress and the test scores of the children; and computer tests to determine the children's ability to program, as well as their understanding of the commands and programming concepts. The children and teachers were given preprogramming instruction on the computer in a laboratory setting.

The procedures that follow were completed during the field trial. Students worked on their copies of the learning materials and/or programs of their own choosing during the laboratory period and their own individual time on the computer in their classroom. The appropriate information was recorded weekly on the teacher forms and the student achievement graphs. An anecdotal record of work with the students during the laboratory time describing the interaction of students with their computers and the learning materials was kept. A final teacher evaluation form and an attitude survey were developed for use after completion of the field trial. Every child wrote an original sample program during the last computer laboratory session. An additional exposure to PILOT by a self-selected group of students attending a computer afterschool program was provided.

After the field trial other procedures were completed. Teachers filled out the final evaluation form describing the computer program as it existed in their classroom, as well as the effectiveness of the program and the learning materials. All students were given the attitude survey. Data were collected and used to evaluate achievement and provide a framework for discussion of the original questions.

Plan of Report

In Chapter II, literature related to the issues within this report will be reviewed. The relevancy of computer programming in the schools, as well as a focus on the languages suitable for children in the early elementary schools and related research will be discussed. Existing curriculum and materials designed to assist in the learning of these languages will be examined.

Chapter III will describe the learning materials and other instruments created for, or used in this study. Special focus will be on background information, features, and contents of the learning materials. The other instruments will be examined in light of the particular study question for which they were designed. The instruments discussed include the teacher form, final teacher evaluation, questionnaire for parents on computers in the home, anecdotal records, multiple-choice computer tests and programming tests, individual student achievement graph, classification schema for evaluating sample programs, attitude survey, standardized reading and mathematics tests, and student data forms.

Chapter IV will describe the site, sample population, procedures, issues and problems in the field trial of the learning materials. Chapter V will present the results of the study based on the original objectives of the materials and investigative questions. Chapter VI will present a summary of the study and its findings, explore implications for learning and the classroom, and propose recommendations for future research based on those findings.

Chapter II

LITERATURE SURVEY

This chapter will lend support to the theses that the study of programming by children is justified in the benefits reaped; assistance in programming is provided by more knowledgeable others or specially designed materials; and case studies, projects, and related research provide much insight into programming demands and individual gains. The sections that follow will look at the claimed benefits of turtle-graphics languages, especially the thoughts and work of Papert and others of the M.I.T. Logo group; the findings of work with children in Logo from case studies and projects; the existing research in the area of cognitive gains and understanding of programming; and the curriculum materials created for learning programming.

Research results have been inconclusive in supporting some of the claims of programming in a turtle graphics language but have provided many insights into the learning of such a language. Greater depth of understanding of the issues and findings can be found by reading Papert's Mindstorms for the philosophy behind Logo learning, Watt's "Logo in the Schools" for the results of related projects and work with children, and Pea and Kurland's "On the Cognitive Effects of Learning Computer Programming: A Critical Look" for a review of the existing research and study needs for the future. The research that has been done in this area has centered around Logo, although what has been shown, may in most instances, apply in general to programming with

young children. These studies will serve as a basis for guidance and decision-making for future research in implementing Logo and/or other turtle graphics languages into the elementary school classroom and the expected gains to the individual in doing so.

Turtle Graphics for Young Children

Many reasons have been proposed for introducing the turtle graphics of the programming language Logo to young children; in fact, Logo was created with children programming in mind. Papert, the creator of Logo, believes that "the child's intellectual growth must be rooted in his experience"[1] and that the computer offers a chance to provide an environment in which the child can become

> highly involved in experiences of a kind to provide rich soil for the growth of concepts for dealing with thinking, learning, playing, and so on.[2]

He also contends that the computer offers the child the opportunity to become involved in projects lasting long enough "for the child to become personally—intellectually and emotionally—involved."[3] In addition to enhancing a child's intellectual powers and involvement in his/her experiences, Papert feels that Logo offers the child an opportunity to do mathematics in such a way that the creativity that should be a part of mathematics is present. Logo may provide beginners with a space in which to enjoy this creativity. The computer can thus serve as a vehicle for mathematical explorations.

[1] S. Papert. "Teaching Children Thinking" MIT Logo Memos (1971a), p. 4-1.

[2] S. Papert. 1971a, p. 4-1.

[3] S. Papert. 1971a, p. 4-3.

In fact, Papert states that the Logo language was "developed expressly for the purpose of teaching children, not programming but mathematics."[4] Perhaps through an environment that allows for mathematization, the child will see mathematics in a more global as well as a more creative sense from the very beginning.

The intellectualizing, the commitment to long term projects, and the rich potential that Logo offers to the user for exploration of mathematics in a creative environment, are possible due to the characteristics of the language itself, namely the turtle graphics components and the rich procedurally-based component that allows for expansion. The turtle graphics elements provide instant visual feedback on the screen thus giving key words (commands) in the language a recognizable characteristic. The simple programs created, often with a limited number of programming structures, can provide hours of explorations. The idea of a computer-based microworld, conceived by Papert, is one central to this exploratory nature of the language. A microworld, according to Papert is

> a subset of reality or a constructed reality whose structure matches that of a given cognitive mechanism so as to provide an environment where the latter can operate effectively.[5]

The computer allows for the creation of such worlds to be constructed and explored. The turtle itself provides for an unlimited number of microworlds; one of those being the creation of a circle. The child is able to assimilate body knowledge from walking out the circle, to new

---

[4] S. Papert. "Teaching Children to be Mathematicians vs. Teaching About Mathematics" MIT Logo Memos (1971b), p. 7.

[5] S. Papert. "Computer-Based Microworlds as Incubators for Powerful Ideas" MIT Logo Memos (1978), p. 1.

mathematical knowledge about the formation of the circle by the turtle on the screen. Microworlds provide in their expandability an avenue for mathematization of the world of turtle graphics. Relationships between shapes drawn on the screen can be rediscovered and serve as an entry into more mathematics.

Watt sees the integration of turtle graphics into the elementary school students' educational environment as a particularly fruitful experience for the individual.

> Within the universe of Turtle Geometry there is room for different students, working individually, to create their own sub-universes or microworlds with their own limited but expandable set of concepts, and related activities and projects. The pedagogy of the LOGO classroom can be seen as the task of helping each child to create, explore, and extend his or her own microworlds.[6]

The power of Logo to create an environment for the growth of intellectualism, the involvement in meaningful projects, and the exploration of mathematics should be investigated further.

Case Studies

While a case study is limited in the type of information helpful for classroom management of such an individualistic activity as programming, the insights gained from working with a child intensively helps identify areas of concern, avenues of success, and growth possible. Working intensively with many children individually for long periods of time often reveals ways of introducing the material in such a way that individual autonomy, style, and creativity can be preserved. Solomon finds through her work with individuals that she has developed

[6] D. Watt. "A Comparison of the Problem Solving Styles of Two Students Learning Logo: a Computer Language for Children" Proceedings of the National Educational Computing Conference (1979a), p. 256.

models for introducing people to Logo. She attempts to get the individual to do something with the computer that is familiar to him/her although not through a computer experience and then build on that in the next experience with Logo.

> Flexibility is one of the most powerful ideas in this culture, but to be flexible implies having a model to depart from. Thus, I have a model in mind of paths a beginner might take in a first session.[7]

Solomon and Watt found that in working with children a variety of learning styles was evident thus necessitating different strategies for working with them. Solomon details these styles: planner, macro-explorer, and micro-explorer. The planner developed a well thought out plan for his/her program in either a top-down or bottom-up fashion. The result may be as originally planned. The macro-explorer tends to experiment with subprocedures in order to create a finished product rather than to begin with a specific plan. The result in this case has a wide range of possibilities. The micro-explorer seems to need time to explore small pieces of information often in a repetitious manner. The results may be only to assure themselves that something is indeed as it seems. Solomon and Watt found any child may possess all styles of learning described. Understanding these styles may help in the development of strategies for working with each learner; it may help us see when a goal or further exploration might be suggested or planned for or when a shift from one mode to another might be appropriate.

Solomon, in her work with a seven year old girl, was able to

---

[7] C. Solomon. "Introducing Logo to Children" BYTE 7 (1982), p. 198, 200.

provide many insights into assistance given to a child programming in Logo. Solomon proposed changes to a program or in the use of a procedure in order to provide the child with an opportunity for extending or testing what she knew, as in the take-this-procedure-and-see-if-you-can-make-this activity. She was also able to keep the child on the right track or pose a question that helped her see what she needed to do in order to solve some programming problem. Thus intervention was used to provide a challenge, ask a question to force the child's awareness of some programming concept or procedure, help the child realize that what she needed to do she already knew how to do, and offer suggestions for program management. Increased confidence and independence were seen to be the results of working in this way; it is unclear the exact part that increased use versus assistance when necessary played, although it may be possible that both are helpful and/or necessary for attaining programming independence.

Watt's work, as part of the Brookline Project discussed in greater detail below, entailed following the programming experiences of 16 students during the first year of study in the project. While he did not work with only one student at a time, he did follow closely the work of the sixteen children who were part of the project. Logo seemed to provide the setting in which all children could grow. For example, Deborah, one student in the study, felt quite insecure at the computer and through restricting the number of commands she allowed herself to use, was able to produce a subworld of Logo in which she could confidently work. She was later able to break out of this subworld and use the other turtle graphics commands. The flexibility of Logo and the

program allowed her to take only a piece of Logo and acquire a sense of programming and her ability to function in it. In looking at all 16 students, differences in the amount of planning done and changes in initial plans, amount of explorations and visualization, and ability to manipulate the turtle were observed. Each child's combination of the above characteristics was unique. Accepting a child's initial planning scheme, realizing that it may be bringing to the individual a sense of his/her role in controlling the turtle and the task of programming, was helpful again in giving a child a positive feeling about his/her computer interactions. Having a sensitive and Logo-knowledgeable teacher willing to accept children's explorations was also crucial to their growth in the understanding of structures and the advantages in planning in a certain way.

Case studies and intensive work with children while programming in Logo have offered many useful insights into introducing programming to children and helping them become confident, independent programmers. A sensitivity to the different learning styles in programming allows a teacher to remain flexible in an environment that lends itself well to this flexibility. Intervention during programming may be to pose a question, present a challenge, or to help a learner move toward a more productive way of working. Despite the differences in individuals with regards to planning, visualization, and control of turtle movements, children can be guided in an appropriate and meaningful way when programming so that all have rich learning experiences.

Projects/Related Materials

The Edinburgh Logo Project was conducted to determine whether programming in Logo might affect the mathematical abilities of twelve year olds in situations where their grasp was weak. Although an adapted version of the Logo language was used, a more structured approach to the teaching of Logo than is currently promoted by Papert was adopted. A materials approach was used in which 33 pages of structured worksheets introduced students to

> computational ideas such as procedures and sub-procedures, variables and recursion; problem solving tactics like decomposing a problem into parts, and the use of debugging skills such as running a trace facility to get a commentary on the execution of a procedure.[8]

Each worksheet, developed for individual, self-paced study, states its purpose, gives a sample Logo procedure to be typed in, and follow-up exercises suggesting modifications to the procedure or adaptations to another problem area. Great care was taken to provide a model for computer concepts with the use of analogies.

While the project did not yield any significant results in regards to greater mathematics achievement, many insights into programming and stages of learning both concepts and programming were observed. The concept learning seemed to reflect their choice of options in the worksheets themselves, as the student 1) copied and ran a procedure containing the concept, 2) modified it to make an "isomorphic" one, 3) altered the procedure's structure, and finally 4) employed the concept when needed.

---

[8] J. Howe. "Developmental Stages in Learning to Program" Cognition and Memory: Interdisciplinary Research of Human Memory Activities (1980), p. 2.

Learning stages were observed. While Solomon describes the styles of Logo learners as three types depending on the amount and type of planning and/or free exploration, Howe categorizes the learning stages based on the programs produced by the learner. The first stage is seen as product-oriented in which drawings were produced without much concern to process. Evidence for this is shown by the fact that many children worked in the immediate mode to create rather than write a procedure, extranneous lines were not removed from the program, the paper under the turtle (the floor robotic variety) was moved to create a certain effect instead of changing the computer's command, and other's procedures were copied so as to produce the same effect. The second stage is seen as the style conscious stage where the student perceives that a certain style of working in Logo should be his main concern. This stage was most visible in a student's work in the immediate mode, recording the sequence of commands that produced the drawing and then putting them into a procedure or procedures, not necessarily in the most efficient or practical way. The third stage, the problem solving stage, is characteristic of the student's ability to use the available resources for solving the problem at hand. Activity is directed at understanding the problem rather than changing the procedure. The stages characterized by programming ability might be categorized as exploratory, procedural writing, and problem solving. The author maintains that the child cannot move from the product/exploratory mode until he has incorporated new mental structures into his work upon which he can build his concepts. Their use of a model whenever appropriate in the curriculum materials to illustrate a concept is

intended to assist in that move.

The Brookline Logo Project involved 16 sixth grade students with a variety of academic abilities and interests. The adult who worked with the children had an expertise in Logo as well as many years of teaching experience. The lab held four computers allowing for four children to be working in the lab at any one time. Children worked with a computer for approximately 20 to 40 hours during the year. An attempt was made to look at what was learned by children programming in Logo, the relationship between learning mathematics and programming, the learning styles used, and the kinds of programming choices made. The program was built around projects of the children's own choosing although a certain degree of structure was built into the beginning work in Logo. All children were given some common introductory material with a stress on turtle geometry projects at the beginning of the study period. The children were expected to work through these learning materials although

> deviations from the plan were allowed when it became clear to the teacher that this would lead to a more meaningful learning experience for the student and them.[9]

As the year progressed the students took up projects of their own choosing with the adult offering guidance as necessary.

As in the Edinburgh study, the Brookline Project results tended to show subjective findings rather than concrete evidence of knowledge obtained or transferred or benefits gained by a Logo group as compared to a non-Logo group, although some evidence was shown that working with

---

[9] S. Papert, D. Watt, A. diSessa, & S. Weir. "An Assessment and Documentation of a Children's Computer Laboratory" Final Report of the Brookline Logo Project (1979), p. 1.14.

turtles, leads to a measurable improvement in the ability to estimate angles. It had been supposed that working in Logo with turtle graphics would improve one's grasp of geometric ideas and in a more concrete manner since turtle geometry is more related to body geometry or mental body image. While this transfer may have been intuitively expected, another transfer was seen, that of the transfer of feelings. Some of the students displayed very strong, positive feelings towards their work with the computer. This is not a cognitive gain, but may be important to note nonetheless.

It was shown that the Logo learning environment was suitable for many different kinds of students, thus being the impetus for another NSF project designed to have learning disabled and physically handicapped children learn Logo with subsequent positive findings in what Logo brings to these children in terms of satisfaction in learning. No standardized tests were issued and the problem solving and mathematical tests created by the staff of the project showed only inconclusive results.

It was found that the project in order to be successful in helping children grow in their working knowledge of Logo needed "an extremely sensitive and knowledgeable teacher, with a great deal of time to consider the needs of each student."[10] While it was hoped that the report of this project would serve as a resource to other teachers or schools without this expert in Logo and teaching, in fact, one result of this project was to realize the need for designing curriculum materials, the emphasis of the second phase of the project.

---

10 D. Watt. "Logo in the Schools" BYTE 7, p. 120.

For the second year of the project in which teachers were given a small amount of training, materials were developed for both teachers and students. Each classroom had the use of a computer for 8 to 12 weeks during which time the children worked in pairs or alone at the computer while the rest of the class followed their normal classroom routine. To augment their classroom work, they met each week for a lesson and idea session. The curriculum materials consisted of introductory materials for students in grades four through six and some advanced Logo projects built around the "dynaturtle" games which the children were able to play and modify.

One especially exciting outcome of the project was the emergence of students in the roles of teacher and expert. Cooperation at the computer was especially high and rewarding and formed a basis for much of what was learned by other students. According to Watt

> It had been assumed at the start that teacher knowledge would be a major limiting factor in what the students could achieve. It turned out that this was not the case. The limitations on student knowledge were what limited what other students could learn.[11]

However, the strength, knowledge, and support of the NSF staff in developing sound curriculum materials and providing assistance when needed cannot be overlooked in this assertion that programming can be taught with fairly unknowledgeable teachers. This situation was successful in getting children to program, but the support provided may have been the factor most responsible for that success. As was noted earlier, Watt himself remarked that the importance of the knowledgeable expert to the learning of the children in the first year of the study

---

[11] D. Watt. "Logo in the Schools" BYTE 7, p. 126.

was crucial. The success of a project to get children programming may depend on the expert assistance of a knowledgeable teacher or the ability of materials to provide that expertise.

Curriculum materials provide the guidance that an expert teacher may in other instances. Suggested and student selected projects provide the flexibility necessary within a program. While slight gains in angle estimation were seen in the Logo programmers and a positive change in attitudes towards work with computers was noted, no other evidence of transfer of knowledge was shown in the projects described. A Logo environment was felt to be an especially learning-rich one for students of all abilities.

Other Research

While many suggestions for working with children learning to program were given, the Brookline and the Edinburgh projects lacked any concrete results on the cognitive benefits of programming in a turtle graphics language as claimed by the original developers of the language. While Papert, Watt, and Solomon have not substantiated these claims, the Logo pioneers and others feel convinced that from their observations the benefits to young children working in Logo are great. Little has also been shown as to what cognitive prerequisites may in fact be necessary for a child, new to programming, to possess in order to be successful at programming. The inadequacies in the research to date have been discussed in reports from the Center for Children and Technology at Bank Street College and will be summarized here.

Pea and Kurland, from Bank Street College, feel that more research

is needed to substantiate the claims by Papert and others regarding the feasibility of Logo to produce cognitive growth in both programming and mathematics. Documentation of what is learned during programming as well as the cognitive prerequisites to such learning needs to be explored. In his own studies with children (8- to 9- year olds, 11- to 12- year olds), Pea explored the purported claim that working in Logo would improve planning skills. The group of children who had worked in Logo and a control group with no Logo experience were given a classroom chore-handling task prior to and after the year's experience in learning Logo. The task allowed children opportunities to devise the shortest plan for carrying out a series of chores. It was felt that this task was comparable to the series of revisions made during programming. It was found that the Logo group did not score higher on a planning task than a non-Logo group. Judgement on this may be a bit premature however, since the group in question worked on Logo for a year in a fairly nonstructured environment where no evidence is given that any attempt was made to work on the children's programming planning skills. It was clear from the Brookline study that intervention and assistance while learning was helpful in changing the planning style of the Logo learner. To measure the transfer of a skill that was not clearly reinforced, only supposed to be a result of working in Logo, seems inadequate.

While no transfer of planning skills was shown, Pea feels this may be supported by recent findings in cognitive science and observations of the children in the study programming. These findings show that "transfer of problem-solving strategies between dissimilar problems, or

problems of different content"[12] are very difficult to achieve and that even computer science students with several thousand hours of programming work had "great conceptual difficulties in understanding how even brief programs are working".[13] Pea observed little preplanning in program development in the Logo programmers. Most children seem to find a goal and the means to achieve this goal after they had begun programming and had some result on the screen. The goal was based on revision of the existing work rather than starting with a goal and finding a means with which to accomplish it. The presence of programming stages seem absent in Pea's discussion, leading one to suspect that the method of Logo introduction varied greatly from that of Solomon/Watt and Howe.

It has also been observed that transfer of skills, such as planning, may occur only after a substantial amount of programming experience, more than the amount that is able to be learned by children in a school setting. Perhaps this view is most useful as one looks at programming in the elementary school. Programming may not provide for the transfer of problem solving skills at a testable level, but the problem solving experiences, the opportunities for mathematical explorations and creativity, and the actual work with computers in which the learner is in control of the machine, may be shown to be a valuable enough experience to support programming in the elementary schools.

Other information from the same study conducted by Pea and Kurland is useful. In an attempt to study the level of programming expertise

[12] R. Pea. "Logo Programming and Problem Solving" Technical Report Number 12 (1983), p. 26.

[13] R. Pea. 1983, p. 30.

the children had acquired by the end of the year, three 45 minute tests to check on command understanding, the ability to write programs to achieve a stated effect, and the ability to find program bugs were administered. It was found that older children knew more, boys had spent considerably more hours in computer time than girls and had scored higher on the programming tasks, and the command comprehension for the whole group was low. From the test for the six best programmers on depth of understanding of programming commands, it was seen that while certain structures such as conditionals were used, that understanding of those structures was not present. Rote chunks of programs were used, but not understood. Perhaps as might be shown by the Edinburgh Project reports, this may just be a stage in the development of Logo programming skills. The understanding of these chunks may come at a later stage.

Pea and Kurland's examination of cognitive constraints provides some insight into the usefulness of developmental level on learning to program. While Piaget's stages of development - preoperational, concrete operational, and formal operational - may be useful in examining certain types of learning experiences of benefit to the child at a given age, programming skills seem too specific to apply to these general stages. Pea and Kurland suggest that

> there is strong evidence that the development and display of the logical abilities defined by Piaget is importantly linked to content domain . . . to the eliciting context, . . . and to the particular experiences of individuals.[14]

It is just not clear at this point in time how learning to program may

---

[14] R. Pea and D. Kurland. "On the Cognitive Effects of Learning Computer Programming: A Critical Look" Technical Report Number 9 (1984), p. 21.

affect children's performance within their developmental level as defined by Piaget. Since programming has not yet been defined in terms of its component skills, assigning programming a place within a developmental schema would not be possible.

Perhaps as is suggested in the Final Report of the Brookline Logo Project, rather than looking at whether children at the elementary school level can or cannot program, it might be more fruitful to ask what kinds of programming are able to be accomplished by various groups of children. Rather than discussing cognitive level needed to program, the question may be to find those programming activities that young children are capable of accomplishing. Perhaps the need for research to determine what constitutes meaningful programming experiences for a given cognitive level are necessary to begin to study the question.

If programming is to be introduced in a classroom or school without the children having access to an expert in programming in very small groups, it may be necessary to devise some curriculum materials to assist the regular classroom teacher in his/her work with her students. The materials must be flexible and be able to allow for open-ended activities while keeping in mind the types of learning styles and programming stages the children are likely to pass through or remain in without proper guidance. Teacher training and/or creation of materials carrying models for programming structure understanding should be considered. So that one group of children will not have unfair advantage in their learning by the sheer fact that they have spent a longer period of time working on the computer, efforts must be taken to insure all equal access at the computer. Care must be

made that all children who use advanced structures in their programming work have opportunities for exploring the meaning behind those structures so that "chunks" of the language are not used without meaning. Research must look at what skills are necessary to learn programming as well as to look at the learning outcomes of programming in terms of the skills themselves and their possible transfer to other areas of learning.

Curriculum Materials

The Edinburgh materials, described previously, presented a fairly structured approach to Logo learning as compared to the Brookline materials that provide a less structured approach allowing for individuals to work on their own projects as a means of acquiring further Logo knowledge. The curriculum materials developed in the second year of the Brookline study for use by students were a more step-by-step introduction to the computer, Logo commands, programming problems, and suggested projects. After introduction to procedure writing and the Logo editor, work with certain shapes was developed with follow-up projects that were extensions of the procedures just created. Sample projects involving subprocedures and superprocedures were explored in greater detail and suggestions for projects that involved more planning were incorporated into the materials. Recursion was introduced in the same way. Both the Brookline and Edinburgh materials were developed for middle schoolers. While the activities or developmental ideas may be suitable, or at least adaptable for use with younger children, the reading level of the materials is too high.

Other materials available during the development of the PILOT

learning materials for this study included a TI Logo Manual. It is descriptive in nature but contains an actual sequence of activities geared more towards an older learner or teacher. The activities are very structured. Cue cards were developed to provide the young user with a simplified set of instructions for working at the computer. The TI Logo Curriculum Guide is written for the classroom teacher. There were manuals available for other turtle graphics languages such as PILOT and Kidstuff which were intended for use with older users. After the PILOT materials were written, other learning materials for students were created, again for middle schoolers. The Education Collaborative for Greater Boston wrote a Logo Curriculum for intermediate grades that provide activities in a very structured set of ordered worksheets with very little provision for open-ended exploration. The existing materials, however, provide insight, ideas, activities, and sequence for the creation of other materials.

Chapter III

DESCRIPTION OF LEARNING MATERIALS AND OTHER INSTRUMENTS

Learning Materials

Background Information

The materials described in this section were created to provide second and third graders with the means to learn elements of the computer programming language Atari PILOT, a turtle graphics language similar in focus to, but procedurally different from Logo. While the children in the study would be working in the computer lab for a thirty minute period each week, they also would be spending an additional thirty minutes working independently on the computer in their classroom. Their teachers were just beginning to program in PILOT and for the most part were not available to work with the children during their in-class computer time. For these reasons, materials were designed through which the Atari PILOT language could be learned independently or with a minimum of assistance. It was intended that the materials provide a springboard into learning the computer language; that children would explore the language initially through the materials and, as time went on, begin to explore on their own. It was not intended that children do only what was in the booklet as in a reading workbook.

The learning materials include elements that embody in them the specifics of PILOT, with attempts made as early in the materials as possible to give the user a sense of the power of a computer language.

Reiteration, looping, conditionals, creation and use of modules (subroutines), sequencing and the importance of order, preciseness of language, branching or selection, logic structures, and the necessity of structure in programming are all general properties of a computer language that would be included at the appropriate time in the materials.

Reading level of the materials was indeed crucial since the intended users were seven and eight years old, many of whom were beginning readers. It was necessary that words printed in the materials were ones that for the most part would be typed into the computer and that all other important messages either be implied, observed through computer use, read from very simple phrases or sentences, or sent via pictures or drawn screen displays.

The children were given opportunities to become familiar with the keyboard and some of the PILOT commands before they worked on the programming materials in order to eliminate the need for assistance with relatively minor machine and/or language-specific problems. For that reason, work in PILOT in the immediate mode was given for a few weeks prior to their work on programming. Materials for this preprogramming work were designed in a format similar to that of the programming materials. The children worked on these for approximately two weeks.

While many sequences of programming commands and structures are possible, one course was chosen that would be followed by all. The study was designed to look at the self-pacing/individualized capabilites of the learning materials. In order that comparisons of

learning materials completed be carried out, all students needed to work on the same materials, in the same order. An attempt was made to thoroughly know the programming language PILOT before beginning to write the materials so that a rich and meaningful development of structures and commands would be presented.

Imperative in creating any materials to be used with a learner as his/her introduction to new ideas, is a sense of what that body of knowledge and understanding can bring to the individual. While these materials had to introduce language that could not be invented, they also had to take into account the wealth of the language and the ideas embodied in it. PILOT has the potential of offering the learner rich experiences in logical thinking, mathematics, communications skills, problem solving, and decision making.

Contents

The learning materials are divided into units or parts. Each part introduces new commands and programming ideas, building on information already known or introduced in previous parts of the materials. Each part contains two or more subparts called sections. The first part was written prior to the field trial. Subsequent parts were written as needed so that changes to form or content, based on observations of effectiveness, would be possible. (see Appendix A for a complete copy of the learning materials)

Features

The learning materials are consumable booklets designed to be written in by the user. While the general appearance is that of a

workbook, the activities have been designed as continuous ones, each building on the knowledge and understanding of the previous activities. The user works on the activities at his/her own pace. For this reason the amount of material to be read has intentionally been kept to a minimum. Much of the printed material is information to be typed into the computer. Observations are to be made and recorded, questions are to be answered, and programs are to be completed, changed, and/or created to achieve stated effects. Users, through observations, are to study the effects of programming structures or commands. Practice occurs when the learner is asked to use a command or structure. The learning materials are divided into three parts; each part being further divided into two or three sections. Each student received Part 1; Part 2 was issued upon completion of Part 1, and so on.

Some features are print-specific. Programs or commands are printed and expected to be typed in. This affords the students practice with finding the keys and a sense of the order of commands, numbers and so on. When the user types in a new command, his/her attention is focused on this word and its connection to what happens next on the screen.

The print used in the learning materials was of two types; the larger type indicated the words and numbers that were to be typed into the computer and the smaller type indicated words and numbers to be read for information and instruction. For example,

```
10 GR:TURN 120
RUN
```

would be typed into the computer, while

Change line 20

Run the program

would be read for information.

The REPEAT command is introduced early, so as to shorten typing time. Earlier work with children of this age revealed that finding keys often proved to be a problem. Illustrations are used throughout, often as replacements for words, to show necessary screen displays, or to illustrate something happening within the computer, program, or command. General computer language is used whenever possible. Examples include the following:

Run the program.

What do you see on the graphics screen?

Some features are response-specific. The user is asked to draw some conclusions, answer some questions, and draw some pictures based on what happened on the screen, and to record his/her answers, pictures, and programs, so as to provide some written record.

Draw a picture of the run.

What is missing from the program?

There are THINK! questions interspersed throughout the materials to ask the children to think about what has just happened. This is an attempt to have them stop, ponder, and perhaps hypothesize about what has happened instead of just continuing on.

What happened when you took out line 30?

Programs are examined to determine the effect of certain changes on the outcome of a run of the program. Skeletal programs with a few or all of the commands missing are included to test the learner's under-

standing of the commands and structures of the language introduced up to that point. Programs that are extensions of existing ones are also requested. The following lines are indicative of the above features:

Put a star before the correct line number.

Change the program so that both shapes are larger.

Make a program with a flashing star.

Make a circle that looks like this one.

Some features relate solely to programming. The turtle graphics commands, those commands that allow the user to create graphic displays on the computer screen by moving a turtle (invisible in PILOT), as well as system commands (LOAD, RUN, LIST, etc.) are introduced by asking the student to type them in and observe what happens.

Editing features are introduced as necessary. The user learns by doing and observing the ways to change, delete or add a line to an existing program. Deleting lines from a program is one way to see the effect those lines have on the program. In this way experimentation, as a vehicle for exploration, is encouraged.

The subtle differences of commands are explored through the introduction of both in the same setting. For example, GR:QUIT clears the graphics screen and returns the user to the text screen. GR:CLEAR clears the graphics screen but keeps the graphics screen visible.

Commands that have a numerical component are experimented with so as to provide an understanding of relative quantity within those commands. For example, changing the PA: (pause) number gives the user a feel for length of time associated with the number.

Programming structures and commands necessary to those structures

are explained through pictures and words, and explored through observation of changes on the screen as a result of changes in the program. For example,

Add these lines .

```
5 *LOOP
70 J:*LOOP

RUN
```

provides the user with his/her introduction to, and observation of loops.

Other Instruments

The instruments used in the study will be examined in light of the research question and/or primary objectives for which they were designed or used. Each question will be stated here followed by a discussion of the instrument(s) used to investigate that question.

Question 1: In regards to learning materials completion, what is the a) achievement level, b) extent of self-pacing achieved, and c) kind and amount of assistance given?

To evaluate student achievement on the accuracy of completion of the learning materials the following objectives will be used.

The student will be able to:

1. use the appropriate PILOT commands.
2. draw conclusions from computer observations.
3. add to and/or delete lines from a program.
4. edit given programs to achieve a given effect.
5. fill-in missing program parts to create a given program.
6. use appropriate angle size or length size.

7. create programs to achieve a given (stated) effect.
8. create programs of his/her own.
9. label program lines by the action performed there.
10. fill-in missing elements on a chart.
11. draw conclusions from a chart.
12. use a counter and a conditional in a program.
13. use programming structures introduced.
14. edit his/her own programs.
15. use the materials as a reference.

In order to evaluate students' understandings of the information presented in the learning materials, two tests were devised: Multiple-choice Computer Tests and Programming Tests. The Multiple-choice Tests are administered individually to a child at the computer upon completion of each part of the learning materials. The child is asked to type in the information on the chalkboard, which is a command to the computer to load the test program. All information is given to the child via the computer. If there appears to be difficulty in reading the information, the examinee would read the question, clarifying when necessary. It is the intent here to measure only understanding in programming. Every attempt is made to have the children feel comfortable with the testing situation. At the end of the program the child's score appears, indicating correctness of each item. Each test contains six to eight items to evaluate the principal objectives of the Multiple-choice Computer Tests that follow:

The students will be able to:

1. match simple program listings to program runs.

2. identify which PILOT command or computer key has been used.
3. use the appropriate computer commands and keys.
4. identify the program lines by the action performed there.
5. demonstrate understanding of PILOT commands and structures.

(see Appendix B for sample screen displays for each test item of the Multiple-choice Computer Tests)

A Programming Test is administered immediately after a Multiple-choice Test to check on programming abilities and understanding. They require the student to write a short specific program. The student is given as much time as necessary to complete his/her program. The Programming Test for Part 1 is "Write a program to make a shape"; Part 2 is "Load the program STAR7. Then change the program so the star flashes"; and Part 3 is "Write a program that uses a counter".

A form, the Individual Student Achievement Graph was constucted for each student. His/her achievement on the computer tests and the completion dates of parts and/or sections of the materials was recorded on this form. This form would later serve as a source of information for a more complete individual student profile, the Student Form.

The Individual Student Achievement Graph consists of three distinct parts: two tables and the primary component, a graph. The first table, with column headings "Date" and "Section Completed" is a place to record the data as it was received. The graph is a pictorial representation of the same information with "Parts/Sections" as the vertical component and "Dates" as the horizontal one. (see Appendix C for a sample Student Achievement Graph)

To evaluate the kind and amount of assistance given two instru-

ments were used: the Teacher Form and the Anecdotal Record. The Teacher Form contains information recorded by the teachers during the students' in-class computer time describing student difficulties with the materials and the computer, and would be used to evaluate the material's effectiveness. While the teacher would be present when each student was working on the computer, she/he would be able to provide only limited assistance due to other responsibilities within the classroom. A teacher reporting form on which classroom teachers could record information concerning the use of materials by their students, amount and type of assistance given, amount of individual computer time allowed each week, and significant problems observed was created. The recording instrument allowed for comments or marks to be quickly made to indicate the kinds of difficulties students were having. To prevent recording inaccuracies, the teachers would fill in the form immediately after assistance was given.

The weekly teacher form requests responses in five areas. The first area, "Teacher Assistance", to be filled in by the teacher with a tally mark in the appropriate box, indicates the kind of assistance the student needed. The second area, "Student Assistance", to be filled in at the end of each week, indicates whether or not students had provided assistance for one another and the frequency, in an estimated percentage, of that assistance for that week. The third area, "Work Time", shows the average amount of independent computer work time each child had had that week. The fourth area, "Significant Problems", indicates both teacher and student computer problems. The fifth area, "Other Comments", is filled in when necessary. (see Appendix D for a sample

Teacher Form)

Anecdotal Records were kept on observations of student work during the laboratory time. While teachers would be able to contribute many observations on the students and their work during class time, the laboratory times would also be a rich source of information on student work and materials use. Whenever possible, narratives on these observations would be written to assist in evaluating the success of the materials. No form was created for this purpose. Anecdotal records would be kept on regular notebook paper. The contents would vary. Often a description of a particular incident would be noted; other times generalizations would be made.

Question 2: After approximately six hours of in-class use and six hours of laboratory sessions with the materials, what kind of programs are children able to write in 30 minutes when given instructions to "write an interesting program all by yourself"?

The Classification Schema was devised to provide a framework for categorization and evaluation of the sample programs written at the end of the field trial. This categorization in turn, allowed for comparisons of individuals and groups. The original schema had to be revised to reflect the actual programs written by the students so that all categories in the classification included at least one program and each program fit into one and only one category. The lowest category consisted of those programs that were copied directly from materials and the highest consisted of those that had more advanced programming structures incorporated into them.

Classification Schema for Evaluating Sample Programs

Type 1: Copied program directly from materials

Type 2: Imitated simple program

Simple string of commands

Type 3: Simple adaptation of given programs

Type 4: Imitated more complex program

Simple string of commands with obvious editing

Type 5: Simple original program: sequence of commands with a purpose

Type 6: More complicated original program

Type 7: Advanced original program

Samples of each type of program can be found in Chapter 5 under the discussion of Question 2.

Question 3: To what extent are students' attitudes concerning work with computers affected?

An Attitude Survey was developed in order to evaluate the student's attitudes toward computers and about themselves in relation to their work with computers after using the learning materials. It is a Likert-like instrument with three response choices. For each of the twenty items, the student selects from three faces, the one that best shows the way he/she feels when a survey item is read to him/her. The first face is smiling, the second is passive and the third is frowning. The items are statements, often in command form, such as "Go work with the computer now." so that the students would react as they might had the remark been made to them in the classroom or laboratory setting. The survey measures the following:

1. attitude toward time spent working with computers
2. attitude toward writing a program
3. attitude toward editing a program
4. attitude toward accepting aid from another person
5. self-confidence in working with the computer
6. motivation to begin or continue work on the computer

Items often measure more than one of the attitudes. For example Item 1 in the Survey reads "Go work with the computer now." While a positive response to this shows a positive attitude towards time spent working with a computer, it also shows a high motivation to begin or continue work on the computer. Some items measure only one attitude. Below are the statements in the Attitude Survey with numbers after them to indicate which of the attitudes each attempted to measure.

1. Go work with the computer now. 1, 6
2. Find the mistake in your program. 3, 5, 6
3. I'm coming to help you now. 4
4. You finished your program! 2
5. Show our visitor what you can do with our computer. 5
6. It is time to leave the lab now. 1, 6
7. Here's something new for the computer. You can be the first to try it. 1, 5, 6
8. Try to write a program of your own. 2, 5, 6
9. Work with a partner today. 4
10. Look at this neat program someone made today! 5
11. Fix your program. 3, 5, 6
12. It is your time to work on the computer. 1, 6

13. You can stay and write another program. 2, 6
14. I'll find someone to help you with your program. 4
15. Work by yourself today. 5
16. Write a program to do something exciting. 2, 6
17. It is time for computer lab. 1, 6
18. Now change the program so it does something different. 3, 6
19. You can stay and work in the lab for a while. 1, 6
20. You need to change a couple of lines in your program. 3

The response sheet has a place for the child's name at the top. There are twenty groups of faces all in the same order from smiling to passive to frowning, numbered to correspond to the statements of the survey. (see Appendix E for a sample of the Student Response Sheet)

The Attitude Survey is an informal instrument developed to look at a rather small element of the study. Teachers, colleagues, and research advisors were asked to group the survey items under the appropriate attitude, 1-6 above. After giving the tests, teachers were asked to look at the scores to check for discrepancies in the rating in order to determine the extent to which the reponses actually corresponded to individual student's attitudes. This served as an informal check on whether the survey had, in fact, measured what it had been devised to measure.

Each child was given the test independently in a room or hallway with only the interviewer present. The child was given the response sheet and the interviewer would then say "I am going to read some sentences to you. I want you to color in the face the way you feel when I read each one." Each statement was read in turn, preceded by a

number so that the child would have a check as to where the next face should be colored in.

Question 4: To what extent is programming ability influenced by sex, grade level, teacher, students' attitudes about themselves and their computer work, standardized reading and mathematics scores, amount of learning materials completed, or time spent working on the computer?

Besides previously described instruments, the scores on the New York State Elementary Reading and Mathematics tests would be used to compare groups of students. The Day School, site of the study, does not give any type of test unless mandated to do so. New York City private schools are required to give all third graders the Mathematics Test and Reading Test for New York State Elementary Schools (Grade 3 Form R 1983). The tests were available for comparison with data from other instruments in the study.

The Student Data Forms were used to organize data collected on individual children to allow for comparisons by attitudes, class, sex, or grade level. The forms would also present a more complete picture of each child, suggesting differences that may exist between two categories of children. The student form consists of five parts, each reflecting data collected on the whole sample. The first part gives information on completion of the learning materials, the second on the multiple-choice computer tests, the third on the programming tests, the fourth on the programming ability rating, and the fifth on the scores of the attitude survey. (see Appendix F for a sample of the Student Form)

The Final Teacher Evaluation Form, completed at the end of the

study, provided teacher feedback on their perceived effectiveness of the learning materials. It was felt that how the computer materials and the computer fit into the classroom was very much affected by how the individual teachers felt about the computer's presence there. This form was devised shortly before the end of the study, as the weekly Teacher Form did not give a clear overall picture of the material's effectiveness, at least from the teacher's point of view. It was also felt that a description of the computer program existent in each teacher's classroom would shed some light on the way in which the materials were used, the effectiveness of the materials under those circumstances, and the relationship of the viewed independence of the materials versus the students' real independence in the classroom. Since it was also felt that the teacher's own feelings about the computer may have directly or indirectly affected the computer use, use of the materials and independence of those materials, the teachers were also asked to describe their feelings about the computer's presence.

The Final Teacher Evaluation Form asks for responses to two questions and a rating scale of the material's effectiveness. The first question asks each teacher to describe the computer program in his/her classroom during the period of the study; the second to describe his/her own feelings on having the computer in the classroom. On the rating scale, the teachers are asked to indicate the effectiveness of the learning materials in the following areas: clarity (to student), clarity (to teacher), ease of use, reading ease, and independence. Their rating was to be shown with an x on a line, with values ranging from 1 to 5, 1 being very effective. (see Appendix G for a copy of the

Final Teacher Evaluation Form)

The Questionnaire for Parents on Computers in the Home was used to determine the amount of contamination to the study by availability of computers, Atari PILOT, and/or other turtle-graphics languages in the home. The effectiveness of the materials or amount of work time on the computer could in part be influenced by contamination of knowledge from other sources. (see Appendix H for a copy of the Questionnaire for Parents on Computers in the Home)

Chapter IV

FIELD TRIAL

School, Sample, and Length of Study

The Day School, an independent N - 8 school in New York City was chosen as the field test site for the study. The school's emphasis is on the child's growth both socially and academically with most academic endeavors centering around exploration of materials and ideas in a fairly informal environment. The achievement levels of the children, while not documented by intelligence and/or aptitude tests, cover the entire range of abilities that might be present in any public school located in a middle class community. Admissions to the school is based solely on the reciprocal offerings of the school and the child and not on academic standings and/or ability. In the sample group, for example, there were some children reading below grade level, a few with learning disabilities, as well as some exceptionally bright children. The sample consists of 58 children; 29 second graders and 29 third graders. There are 25 girls and 33 boys. This sample includes the entire second and third grade population. The field test took place over a twelve week period from the week of October 25, 1982 to the week of February 7, 1983.

Procedures

Meetings were held with the staff prior to the beginning of the study to answer questions on the programming language Atari PILOT and

to discuss the placement of computers in the classroom and matters directly related to the study. The study began approximately one month after the students had had their first weekly computer laboratory session. On the three days of the week when the computers were not in the laboratory, they were in the classrooms, each second and third grade classroom having one. During that month prior to the beginning of the field trial, teachers worked through many problems on flexibility of scheduling and equality of computer access.

Teachers and students were given preprogramming instruction on the computer in the laboratory setting. They were able to investigate some Atari PILOT commands in the immediate mode in the laboratory and classroom for approximately four weeks before receiving the learning materials for the field trial. For the last two weeks of this time, children used preprogramming materials similar in nature to the programming learning materials they would soon be using. Teachers began their PILOT programming instruction, working on their own when possible and/or interested in doing so.

The learning materials were introduced to the children during a computer laboratory period. The children needed to be clear on the two types of print that were used to differentiate between words and numbers to be typed into the computer and ones to be read, as well as the necessity of pressing the RETURN key at the end of each line of print typed into the computer. An attempt was made to observe every student as he/she began to work, as well as when trying something new. For the remainder of the field trial period, the students would work on the materials and/or programs of their own choosing, both during the

laboratory period and their own individual time on the computer in their classroom. Since there were at most six computers in the laboratory at any one time, the students very often had to share a computer. Every attempt was made to find children who were at the same place in their materials to work together. Weekly staff meetings were held to discuss problems arising from the use of the computer and/or materials.

The materials were used in class and in the laboratory for 12 weeks. Children were assisted in class, when necessary, by their teachers or peers. In the laboratory, there were usually two adults present, the laboratory instructor and a teacher and/or assistant teacher. Whenever possible, the children were directed back to their own resources, their previous work or their ability to explore and find the answer. For the length of the field trial, the students used their materials and/or wrote programs of their own in Atari PILOT; no other unrelated work was done on the computers either in class or in the laboratory with the exception of one teacher who did a Thanksgiving and Christmas group computer drawing with his class.

Whenever a child had completed Part I, II, or III, they were issued two tests: the appropriate Multiple-choice Computer Test and the Programming Test for the part completed. These tests were administered to the student at a computer that was isolated from the rest. Each Friday a record of sections completed, tests taken and scores received were recorded on the Student Achievement Graphs.

Both the teacher forms (completed by the teacher) and the anecdotal records (completed by the lab instructor) were attempts to record student problems with the materials. Both were kept on a weekly basis

and were used as a basis for determining how the subsequent parts of the materials would be written, who needed assistance, what clarification might be needed, and how the students might be helped to better understand some aspect of the materials.

During the last laboratory session in the field trial, each child was asked to create a program of his/her own. They understood that they could use their learning materials and ask questions but that the program had to be an independent activity. They were told to take the whole period if necessary. When they were finished, they were to write their program down on the paper provided. The teachers helped write down the lengthier programs at the end of the period.

The Final Teacher Evaluation Form was completed during a staff meeting by each teacher and assistant teacher who had worked with the children. The Attitude Survey was administered to all second and third graders individually in a room or hallway. The teachers also agreed to take the survey. A self-selected group of children took part in a brief additional exposure to PILOT through a computer afterschool program.

Issues and Problems

The field trial was not conducted with a control group. The sample group previously described, used the materials to learn Atari PILOT and was not compared to a group of children learning Atari PILOT without these materials. The intention of the study was to test whether these children could learn PILOT with these materials and the extent to which that learning would be self-paced and independent, not

whether they would learn more using these materials.

The field trial was to be a part of a regular computer program in the site school. For four weeks prior to the field test, teachers watched the interaction between their students and the computer in their classroom, dealt with scheduling issues, and worked toward establishing an equal access situation in their classroom so that these issues would be resolved at the onset of the field trial. They found that flexibility of scheduling allowed them to find enough time in the course of the three days that the computer was in the classroom for all children to have from twenty to forty minutes of independent computer time weekly. Equal access was a crucial issue. In order to guarantee that all children accumulate an equal amount of time on the computer, all teachers devised some sort of scheduling plan. Attention was paid to monitoring this schedule.

During the field trial, there were two significant problems brought up in weekly staff meetings. There were a few students who were refusing their time on the computer. The second problem came in a third grade classroom, where one child, with the availability of PILOT at home, was joined by a few of his classmates in flaunting his expertise. Each problem was resolved and will be discussed in Chapter 5. Children, at times, needed to be directed back to their materials or encouraged to explore independent of their materials. These problems will be discussed further in Chapter 5.

Chapter V

RESULTS

The results of the study will be discussed in relationship to the original investigative questions. An additional section will describe other observations.

Completion of Learning Materials

In regards to learning materials completion, what is the a) achievement level, b) extent of self-pacing achieved, and c) kind and amount of assistance given?

Achievement

Accuracy of completion of the learning materials was examined based on the primary objectives of the materials. While periodic checks were made to insure that the children were working in their materials, all work was not checked for accuracy. Children were to work through the materials at their own pace. It was hoped that if some programming command or structure was not fully understood at the point of introduction, that subsequent work would help clarify that aspect of the programming language.

Based on the principal objectives of the learning materials that follow, the results discussed below were seen.

The students will be able to:

1. use the appropriate PILOT commands.
2. draw conclusions from computer observations.

3. add to and/or delete lines from a program.
4. edit given programs to achieve a given effect.
5. fill-in missing program parts to create a given program.
6. use appropriate angle size and/or length size.
7. create programs to achieve a given (stated) effect.
8. create programs of his/her own.
9. label program lines by the action performed there.
10. fill-in missing elements on a chart.
11. draw conclusions from a chart.
12. use a counter and a conditional in a program.
13. use programming structures introduced.
14. edit his/her own program.
15. use the materials as a reference.

All students (58) in the sample completed Part I, Sections 1 and 2 in which completion of the materials tested objectives 1, 2, 3, and 4. As a group, 71% (41/58) were able to use the appropriate PILOT command (Objective 1), 74% (43/58) were able to add and/or delete program lines (Objective 3), and 79% (46/58) were able to edit given programs to achieve a given effect (Objective 4). Only 60% of the group were able to show evidence that they could draw conclusions from computer observations (Objective 2). While many of the children may well have been able to draw conclusions based on what they saw, writing down these observations may not have been as easily accomplished. In working with many of the children, especially second graders, it was seen that when a question involved writing down more than one word or number or explaining why something was so, the question was skipped.

Most of the inaccurate responses were, in fact, blanks. Many children were just not committed to the completion, or the correct completion, of all exercises. Although every attempt was made to use a limited vocabulary, very often the meaning of what was written was not clear to some of the students.

Twenty-eight students completed Part 1, Section 3 of the learning materials which tested objectives 4, 5, 6, 7, and 8. As a group, 89% (25/28) were able to edit given programs to achieve a given effect (Objective 4), 93% (26/28) were able to fill-in missing program parts (Objective 5), 89% (25/28) were able to use appropriate angle size and/or length size (Objective 6), 86% (24/28) were able to create programs to achieve a given effect (Objective 7), and 82% were able to create programs of his/her own (Objective 8). While all children were able to create programs of their own, again the task of writing down their work into the space provided in the learning materials was just too time consuming for some.

Ten students completed Part 2, Section 3 (Sections 1 and 2 were computer operational instructions) of the learning materials which tested objectives 2, 3, 4, 5, 7, and 9. As a group, 90% (9/10) were able to draw conclusions from computer observations (Objective 2), 70% were able to add and/or delete program lines (Objective 3), 100% (10/10) were able to fill-in missing program parts (Objective 5), 90% (9/10) were able to create programs to achieve a given effect (Objective 7), and 70% (7/10) were able to label program lines by the action performed there (Objective 9). Only 60% (6/10) were able to accurately edit programs to achieve a given effect (Objective 4). It

will later be shown by scores on the Attitude Survey that most students did not like editing their programs. Two of the four students who did not master this objective were students who seemed to feel compelled to complete as much work as possible in their book. Their desire to complete workbook pages appeared greater than their desire to complete them correctly.

Three students completed Part 2, Section 4 of the learning materials which tested objectives 4, 7, and 8. Two students completed Part 3, Section 1 which tested objectives 2, 4, 5, 6, 7, 8, 10, and 11 and Part 3, Section 2 which tested objectives 2, 5, 7, 8, 12, 13. All objectives in these last three parts were accomplished with 100% accuracy.

Objective 14, the student will be able to edit his/her own programs, was not tested for but through observation it was seen that most children could add and delete lines in their programs. As seen in the sample programs discussed later, about 74% (43/58) of the children wrote a middle level or high level program, both showing signs of the ability to edit. In relation to objective 15, the students will use the materials as a reference, an effort was made to have the students find the answer to their questions in their materials where possible. The variability in this use of the materials as a reference was quite extensive. A few kept notes and used the materials often in this way. Many had a difficult time relying on themselves and/or the materials, although every attempt to have them do so was made.

In general, those children who completed more than two sections of the materials, completed them more accurately. Thus, those who got

beyond the first two sections in Part I, generally completed most all items accurately. There were class differences in completion of the materials. In one class, in which only one child finished more than the first two sections of Part I, there were more inaccuracies, or incomplete work, than in the other groups. Another classroom teacher spent more time with those children who were having difficulties, thus more work was completed as their questions were answered. The other two classrooms were more competitive, one of those having many students who seemed to be workbook fanatics; the computer learning materials were seen as another workbook to complete.

There was a significant difference between the amount of learning materials completed by one of the two second grade classes, Class A, and the other three classes in the field trial, as well as between Class A and the other second grade class. Table 1 shows the frequencies with which Class A and the rest of the classes completed the learning materials.

Table 1

Completion of Learning Materials Among Class Groups

	Sections of Learning Materials Completed	
Class	1.1-1.2	1.3-3.2
Class A	13	1
Other Classes	17	27

The χ^2 test for two independent samples was used. For the data in

Table 1, the value of χ^2 is 9.73 which is significant at the 0.01 level. There is a significant difference between the amount of the learning materials completed by the second grade Class A and the other three classes.

Table 2 shows the frequencies with which Class A and the other second grade completed the learning materials.

Table 2

Completion of Learning Materials Among Second Grade Classes

	Sections of Learning Materials Completed	
Class	1.1-1.2	1.3-3.2
Class A	13	1
Other Second Class	7	8

The χ^2 test for two independent samples was used. For the data in Table 2, the value of χ^2 is 5.22 which is significant at the 0.05 level. There is a significant difference between the amount of the learning materials completed by the second grade Class A and the other second grade class.

Children from Class A completed less of their learning materials than both the other second grade and the rest of the classes. While it may have been expected that the second graders would have completed less of the learning materials due to reading ability and/or maturity, it was not expected that one group of second graders would have outperformed the other in this regard. Through observation and discussions

with the staff during the field trial, it was noted that the second grade teachers managed two very different computer programming environments. Class A teacher was very apprehensive about the computer in general. She had no prior computer experience. She felt that her energies had to be divided into other different areas of the curriculum many of which were seen as more important to her and her class. Since this was her first year as a classroom teacher much of her extra time was spent in those areas rather than learning more about PILOT and/or the computer. Although she attempted to help the children, often this was not possible. In contrast, the teacher in the other second grade class was extremely interested in learning as much as possible about the computer and PILOT and had had prior computer programming experience. He worked on two class computer projects, encouraged group and cooperative efforts, was able to work with those children who were reluctant as well as those who were able to move quickly through the materials, and encouraged the children to write programs of their own. Apprehension about the computer and their operation of it remained with the children in Class A much longer than in the other second grade classroom.

The Multiple-choice Computer Tests were designed to evaluate the students' understandings of the material presented in the learning materials. A test was given at the end of each part of the materials. Each test contained six to eight items to evaluate the principal objectives of the tests as follows:

The students will be able to

1. match simple program listings to program runs.

2. identify which PILOT command or computer key has been used.
3. use the appropriate computer commands and keys.
4. identify program lines by the action performed there.
5. demonstrate understanding of PILOT commands/structures.

Each student received a score of 75% or better on each test taken as a whole. Twenty-eight students took Test 1 which tested objectives 1 and 2, upon completion of the three sections of Part 1. As a group, 71% (22/28) were able to match simple program listings to program runs and 100% (28/28) were able to identify which PILOT command or computer key had been used at a 75% mastery level. Three students took Test 2 which tested objectives 3 and 4, upon completion of the two sections of Part 2. As a group, 100% (3/3) accomplished the objectives at a 75% mastery level. Two students took Test 3 which tested objectives 2 and 5, upon completion of the two sections of Part 3. As a group, 100% (2/2) accomplished the objectives at a 75% mastery level. In general, all students who were given tests did well.

Immediately after taking the appropriate multiple-choice computer test, the student was asked to write a specific program. Programming Test 1 asked the student to write a program to make a shape such as a square, triangle, or pentagon; Test 2 asked the student to load the program STAR7 into the computer and then to change the program so that the star flashed; and Test 3 asked the student to write a program that used a counter. All students successfully wrote the program requested. For some, the program was completed within seconds; for others the task took as long as ten minutes.

Self-Pacing

One measure of the success of any materials to be self-paced is the amount of material covered by the individuals of a group in a given period. Since the situation was monitored to assure that all children were using the materials, it was possible to compare the sections completed at the end of the twelve week study to ascertain whether the materials allowed these children to work at their own pace. It needs to be assumed, however, that due to the nature of any group of children approximately the same age, there exists many different rates of working. Therefore it would be expected that with the amount of time on the computer being equal or near equal, the amount of the materials completed would vary from individual to individual to reflect these different rates of working. After approximately 12 hours of on-computer time, the number of sections of learning materials completed by each student, as shown in Table 3, were compiled.

Table 3

Completion of Learning Materials

Section Completed	Number of Students
1.1	58
1.2	58
1.3	28
2.1	10
2.2	3
3.1	2
3.2	2

Table 3 indicates that students were able to complete differing amounts of the materials given the same time frame in which to work on them. Students were able to work through the materials at varying paces. While 100% of the sample (58/58) completed Sections 1.1 and 1.2, 47% (28/58) completed Section 1.3, with 17% (10/58), 5% (3/58), 3% (2/58), and 3% (2/58) completing Sections 2.3, 2.4, 3.1, and 3.2 respectively. This data indicates that to a rather high extent, self-pacing was achieved in the use of these Atari PILOT learning materials.

Assistance

Students did need assistance in order to be able to work in their learning materials. It is not an easy task to separate the assistance given students while working in their learning materials due to difficulties they may have understanding the computer and their interactions with it, from the assistance given due to difficulties in their use of the materials.

The materials were designed to be used independently as a springboard into their work with computers and programming. Interaction, in the form of requested or offered assistance, was given to optimize the effect of the materials. It was not intended that assistance would be needed constantly nor that it be intentionally withheld. If the opportunity arose to present some new idea or problem to solve based on what the student had just done, the opportunity was taken. Questions were asked if it was felt that the children might not be questioning themselves as they worked through the materials.

The students needed to be computer-secure. This was a rather slow process for some. Thus they would need to ask more questions due to

insecurity rather than inadequacies of materials to give them the needed information. It was found that a few students were refusing their time on the computer. In each case, the child was just not secure in what he/she could or should do on the computer. Devoting a thirty minute period to work with that student was most effective. Some teachers would schedule that child's computer time when they would be able to assist.

Assistance in the use of the materials seemed to be of the same kind and magnitude as the assistance needed to begin work in any new area, especially one in which exploration is a part of the experience. Initially more assistance is needed, leveling off to the assistance that is associated with new elements of a subject matter. Since the materials were designed to provide students with the ability to move through them at their own pace, it was likely that more "new" elements would be worked on each week. Also, many of their experiences will be in the area of problem solving. This area, while presenting many challenges, also requires that more assistance be given.

Assistance needed in that first programming session in the laboratory with the learning materials was mostly "reluctance" assistance; the students were reluctant to proceed past a line without first checking to see what they should do next. Most children, however, worked without assistance and/or questions. Since each child was working on the same page in his/her materials that first session, the television screens served as feedback devices to him/her as well as to others in the room. Seeing what someone else had on his/her screen often reinforced the correctness of one's own work. After that first

laboratory session, the kind of assistance needed was that of clarification and understanding of programming procedures and structures. Assistance was also offered to present a challenge or question what the student was doing.

According to the weekly Teacher Forms used to record classroom assistance, children had a variety of difficulties in using the materials/computer. In general, reading difficulties were few. Problems with machine operation lessened as the study continued. Students needed help with a variety of programming and computer problems, as well as finding or correcting their errors. They received help from teachers and/or peers in these matters. Peer assistance became an invaluable aid to some in the classrooms where it was allowed or encouraged; partners were often chosen in two of the classes. The teachers in these classes generally did not feel as secure themselves with the computer and in some cases this forced the students to seek help from another student.

One problem experienced by many but resolved after more programming experience involved the idea of a program as an entity that was stored in the computer memory only as long as the computer was turned on. Students would sit down to work not realizing that in order to continue where they had left off in their materials, they needed to reenter their program into the computer; that the program they had typed in last time was not still there inside the computer.

Most of the children in the study relied on their materials and others in their beginning work with programming in PILOT. The move toward independence came at varying times for the children, but seemed

most related to their ability to make the computer do what it was supposed to do as directed in the materials. Relying on others was often a sign of not being able to take a chance. While many children asked that first day if they should press RETURN after typing in that first program line, most were able, at least by four weeks into the study, to have gained a sense of exploration, a "try it and see" attitude that also carried over into their work with the materials. This ability to explore with each new command or structure was also a sign of greater freedom from others. Other children were very independent in their materials work from the beginning. While the materials were written with as few words as possible, it was clear that at least in the beginning, the better readers relied less on assistance from others. As the less able readers came to know the vocabulary used in the materials, reading seemed to be less of an issue for them.

Children appeared to rely on themselves more as they became comfortable with the computer and their ability to make something happen. The more independent children still looked to others for interaction and assistance, but their needs were different. They no longer needed help with operating the computer or using the materials, per se, but questions on programming continued to arise. They wanted to know how to make the computer do something, whether something was possible, or how to combine what they already knew into what they thought was possible. They became more willing to take chances, experiment, and explore with new commands and structures.

It was found that children often had to be directed back into their materials or challenged to experiment with what they had been

introduced to thus far. In the instances where a child was not using his/her materials, a little assistance seemed to help him/her understand how to work in the materials or what he/she could get out of them. Another kind of dependence was seen in a few children; a dependence on their materials. Their reluctance to explore was obvious. Since it was felt that exploration would insure understanding of the commands and structures in the language, it was crucial that these children do so. Challenge programs were offered with a little guidance. When not electing to do so on their own, children were encouraged to balance the work in the materials with some exploration. Most children did both.

While partners often lessened the need for adult assistance in the classroom or the laboratory, often working with a partner was detrimental to achieving greater independence. A few children did not want to work alone, still apprehensive about their ability to manage on their own.

Programming Ability

After approximately six hours of in-class use and six hours of laboratory sessions with the materials, what kind of programs are children able to write in 30 minutes when given instructions to "write an interesting program all by yourself"?

The last laboratory session of the field trial was given to the creation of individual programs. Each child was placed alone at a computer, given instructions to write a program of his/her own using his/her learning materials folder if he/she desired, and asking for

help if necessary.

The Programming Session

In all cases, except one, the rooms were very quiet as the children worked on their programs. The one exception was Emily's class. Emily constantly talked to herself. Others asked her nicely to be quiet, but she had a great deal of difficulty doing so. She did the same thing during other laboratory periods, but the other talking in the laboratory made it unnoticeable until this day.

It would be useful at this point to describe some observations of student behavior, work, questions and comments during this program-writing session. Generally, children seemed reluctant to ask questions. They had been told that they could ask questions, but that they needed to rely on themselves and their folders for writing this program since it was necessary to see what kind of program they could write on their own. When they did ask questions, they were often referred to their learning materials folders or asked a question that would trigger some prior work. It was rare that a student asked a question that he/she was not able to figure out or reference in his/her materials. Very few children used their learning materials. Most seemed to pull from information within themselves.

Neil asked if he had to do things in his folder or would he be able to do something Charlie had taught him. Throughout their work in the learning materials student interaction, and thus sharing of information, had been high.

A few children turned off the computer at some point and started again. Others achieved the same effect by typing in NEW, something

which had been encouraged over turning off the computer. Some seemed to do this in frustration at not achieving the effect they had wanted, or at least something interesting enough to keep. While most children were generally attentive to the task at hand, they were able to view some of the other television screens if they chose to do so. Pressure to create something meaningful, therefore, was kept strong by the knowledge that others could, if they chose to, view the results of their programs. This fact seemed to be the basis for some of the destruction of programs. Kevin and Scott were very fast typists. They both quickly created programs with some sense of order and a certain degree of pot luck. They hoped that the order they had chosen would produce an interesting result. If the result could not be edited to achieve an interesting result, they would turn off the machine and nonchalantly start again.

Most seemed satisfied with their final result even if there was some frustration in the creation process. Two children cried while working on their programs. Abigail most always created programs with a goal in mind. She knew now what she wanted to do based on previous experience and watching the results of others. Her program plan involved a new idea and she did not know how to accomplish this. She knew that this was something that she could not ask. Matt was genuinely frustrated. An intelligent child, who had missed a lot of school and thus computer time, Matt was in a class that would often work together to create exciting programs. He was well aware of how his work compared with the others and this seemed to paralyze him at this point. An agreement was made to have him come to the laboratory at

another time to work alone with the computer. This seemed to help and he was able to focus on the creation of his computer program for the rest of the period.

Observations made during this session were most important so that information not obtainable from a finished program could be known. The following are four examples of that type of information. Ben T. had used a more sophisticated structure (loop) but removed it since the visual effects were not as pleasing. He was thus able to insert the statements where necessary to create this loop, knowing that often, very interesting patterns would result. This time, however, this was not the case. Peter had made an interesting design using the repeat command. Outloud, but very quietly, he remarked to himself "I think I'll make two of them". He then proceeded to do so successfully. He, like Ben T., preferred his original creation. Thus he eliminated this second design. Ben L. created a snowflake design by repeating a given design, turning a bit, and repeating this process until the design was to his liking. The interesting idea here was that much work went into deciding on just the right number of times to repeat the design and the amount to turn and that this work was not evident in the finished product which could also have been arrived at by chance. Porter would add a few lines to her program and then run the program to assure herself that her latest program line was visible on the screen.

A few children used information from their learning materials. Emily used the lines to create different stars; Justine the same for different shapes. Only three children copied their program directly from the materials. Two of them miscopied one line. They did not go

back and make the correction, although it was obvious that one of the two knew she had made a mistake. The other child was just glad to have something on his screen.

Some students were very relaxed. For the most part, these students seemed to be the ones who were generally more independent in their computer work and more confident in their ability to produce a program that they would be satisfied with. They asked for help, but when they did, were better able to see what kind of help they needed.

Some students seemed to settle for a mediocre program, or at least one that they deemed less exciting than they had produced in the past. Other students had a difficult time settling on a program they considered adequate. This pressure kept them working until they finally created an interesting program or ran out of time and settled on what they had done.

Most students concentrated on the task. Others let themselves be distracted by what others were doing. These students seemed to have added the competitive edge, needing desperately to create a program that was better than the others. Those that did the best were those who could stay focused on the task at hand, had a good idea of what they wanted to do, which was also within the realm of what was possible for them, and had exhibited a fair amount of independence in their previous work with the computer.

Children wrote programs based on information they had obtained from their materials and from their interactions with others. While the programs reflected where children were in their materials, they also reflected what they had learned from each other. Therefore, if

one child had reached the third part of the materials, it was evident that some others not yet in that part, knew about, could use or were experimenting with programming commands or structures from that part. Generally speaking, however, many experimented with and could use some new commands learned in this way, but only one boy was able to incorporate a new structure, learned from a peer who had reached that part of the materials, into his work.

A session like this was able to produce evidence of the programming information the children possessed and were able to use. Although much was shown about the students' programming abilities through observation that a study of their finished products alone could not have revealed, it was also found that in all but one case, this extra information did not change their rating of programming ability. This is very important to note. In these cases, observations were valuable in understanding the children as programmers, yet not in determining the level of sophistication of their programs.

The Classification Schema

The original schema had to be revised to reflect the actual programs written by the students so that all categories in the classification would include at least one program and that each program would fit into one and only one category. Changes reflected a need for clarity (simple original program to simple original program: sequence of commands with purpose), a previously unconsidered category (simple string of commands), or a classification of two types of programs as being at the same level of sophistication. Each time the schema was changed, the programs were again classified until it was clear that

each belonged clearly to one and only one category. Below is the schema as originally created and as revised.

Original Schema	Revised Schema
1. Copied directly from materials	1. Copied program directly from materials
2. Imitated (one from materials; recreated on own)	2. Imitated simple program Simple string of commands
3. Simple adaptations of given programs	3. Simple adaptation of a given program
4. More complicated adaptations of given ones	4. Imitated more complex program Simple string of commands with obvious editing
5. Simple original programs	5. Simple original program: sequence of commands with a purpose
6. More complicated original programs	6. More complicated original program
7. Highly complex programs	7. Advanced original program

Sample Programs

Below are the categories of the Classification Schema for Sample Programs with a sample of each. Comments are included where helpful.

Type 1: Copied program directly from materials

```
10 GR:DRAW 15
20 GR:TURN 120
30 GR:DRAW 115
40 GR:TURN 120
50 GR:DRAW 15
60 GR:TURN 120
```

Line 30 was miscopied by the student; 115 should have been 15. Three children wrote programs of this type; two of them miscopied a number.

Type 2: Imitated simple program

```
10 GR:DRAW 25
20 GR:TURN 120
30 GR:DRAW 25
```

This student spent the entire period typing this in. While it is the beginning of the program that was printed in her learning materials which she had been working on since October, she did not refer to her folder at all. Only two students wrote programs in this category of Type 2 programs; the rest, 10 students, wrote programs in the next category.

Type 2: Simple string of commands

```
2 GR:59(DRAW 45;TURN 1592)
3 GR:TURN -90
4 GR:DRAW 48
5 GR:FILL 79
6 GR:43(DRAW 23;TURN 34)
7 SO:56
8 PA:123
```

The result of line 6 does not appear on the screen. Ten students wrote this kind of program. They seemed to have available to them the programming commands they had learned and at some level an ability to use them. It seems clear in the above example for instance that this boy knew that the sound (SO) command could be followed by a pause (PA) command. It was not obvious from the program whether he knew the effect that the pause command had on the sound command. It seems that he was stringing together commands, without obvious editing. If he had been observing closely what was happening in his program, he would have realized that the figure created by line 6 did not appear on the screen

and would have made some changes in the program.

Type 3: Simple adaptation of given programs

```
11 GR:5(DRAW 15;TURN 144)
12 GR:7(DRAW 15;TURN 154)
14 GR:PENRED
45 GR:3(DRAW 15;TURN 120)
```

Only two children wrote programs like this. Both used information from their folders but combined the information into one program. The child who wrote the program above was well aware of what she was creating and how she was accomplishing it. After the first two line numbers she became aware that she would be able to add more shapes in a similar manner and that she could make the next one a different color. She was making many new discoveries through this assignment.

Type 4: Imitated more complex program

```
9 *SNOWFLAKE
10 GR:5(DRAW 30;TURN 144)
19 GR:TURN 20
20 J:*SNOWFLAKE
```

Of the 19 students whose programs were of this type, only the one above fell into this category, that of a more complex program given in the materials. The others were either a combination of both categories in this type or clearly of the next type.

Type 4: Simple string of commands with obvious editing

```
1 GR:DRAW 34
2 GR:TURN 200
3 GR:PENRED
4 SO:34
5 SO:42
6 GR:60(DRAW 45;TURN 90)
7 GR:GOTO 34,56
8 GR:577(DRAW 62;TURN 144)
```

This student, as with all others in this category, was well aware of what his program was doing and through editing was able to make neces-

sary changes. It was not obvious that he had made plans for something specific to happen from the onset and followed through on that plan or developed any plan after an initial command or two had been written and tried out. That type of planned program was a type 5 program.

Type 5: Simple original program: sequence of commands with a purpose

```
5 GR:PENRED
10 GR:3(DRAW 15;TURN 120)
11 GR:PENBLUE
20 GR:4(DRAW 15;TURN 90)
30 GR:7(DRAW 15;TURN 144)
31 PA:60
32 GR:CLEAR
33 GR:GOTO 15,15
34 GR:GOTO 2,10
35 GR:4(DRAW 2;TURN 90)
36 GR:GOTO 0,10
37 GR:4(DRAW 2;TURN 90)
```

This student, as with the other 14 students who created programs of this type, gave some indication that she had some plan in mind while creating her program. She wanted to create different shapes in different colors, clear her screen and create other shapes in different positions on the screen.

Type 6: More complicated original program

```
10 T:ZAP
20 T:DO YOU WANT TO GET ZAPPED?
25 T:SAY YES OR NO
30 A:
40 M:YES
45 JY:*F
50 T:TO BAD YOU GOT ZAPPEDZZZZZZZZZ
56 E:
60 *F
65 T:YOU ESCAPED THE LAZER!
```

This student knew exactly what he wanted to do and the structures and commands he had to use to accomplish his goal. He displayed an under-

standing of conditionals and the use of the match (M:) command. There were six students who wrote programs of this type.

Type 7: Advanced original program

```
9 GR:PENBLUE
10 *FOOTBALL
20 GR:42(DRAW 15;TURN 22)
30 GR:TURN 15
40 C:#A=#A+1
50 J(#A<10):*FOOTBALL
55 GR:PENRED
60 GR:TURN 91
70 GR:DRAW -45
80 *BASEBALL
90 GR:42(DRAW 15;TURN 22)
100 GR:TURN 15
110 C:#B=#B+1
120 J(#B<10):*BASEBALL
```

Two students wrote programs of this type. Both had Atari PILOT available to them at home and were highly motivated to learn as much as possible about the language.

The Classification Schema yielded three major levels of programming ability described by the kind of programs written. A low level program, type 1 or 2, is a simple program, either a sequence of random commands or one copied directly from another source. While there is some ability to use commands, there is no evidence of editing or a sense of the importance of order. A middle level program, type 3 or 4, is an exploratory program with attempts to use known commands and structures. There is an absence of planning for the most part, but evidence that freedom to explore, the ability to edit programs, and an awareness of what certain commands and/or structures do, is present. A high level program, type 5, 6, or 7, is a planned original program often including a more sophisticated programming structure. There is

evidence of planning or preplanning as well as the understanding of structures. The three levels may be summarized by the terms imitative, exploratory, and planned. The number of programs in each type category of the Classification Schema is as follows:

Type 1: Copied program directly from materials . . . . . 3 students

Type 2: Imitated simple program

Simple string of commands. . . . . 12 students

Type 3: Simple adaptation of given programs . . . . . 2 students

Type 4: Imitated more complex program

Simple string of commands with obvious editing . . 19 students

Type 5: Simple original program:

Sequence of commands with a purpose. . . . . 14 students

Type 6: More complicated original program . . . . . 6 students

Type 7: Advanced original program . . . . . 2 students

Fifteen children wrote low level programs, Types 1 and 2; 21 children wrote middle level programs, Types 3 and 4; and 22 wrote high level programs, Types 5, 6, and 7.

Students are able to write programs at various levels of sophistication. The programs created range from the simple program copied directly from the materials to a rather sophisticated program able to incorporate more advanced programming structures into a program in order to achieve a certain goal, from free exploration to planned programming.

Attitudes

To what extent are students' attitudes concerning work with computers affected?

At the end of the study the Attitude Survey, given to each student, measured the following:

1. attitude toward time spent working with computers
2. attitude toward writing a program
3. attitude toward editing a program
4. attitude toward accepting aid from another person
5. self-confidence in working with the computer
6. motivation to begin or continue work on the computer

In general, students exhibited positive attitudes in all areas measured except editing a program. In both attitude toward time spent working with computers and attitude toward writing a program, most all scores were clustered at the highest possible scores. Children felt extremely positive about their time spent working with computers and writing programs.

In both motivation to begin or continue work on the computer and accepting aid from another person, scores were high with a clustering of scores falling just before the highest possible scores. Children felt very motivated to begin or continue work on the computer and very positive about accepting aid from a another person.

Most scores of the self-confidence scale were in the upper half of possible scores (positive) with the greatest cluster of scores falling in the middle of that half. Children were for the most part self-confident about their work with computers, but not overwhelmingly so.

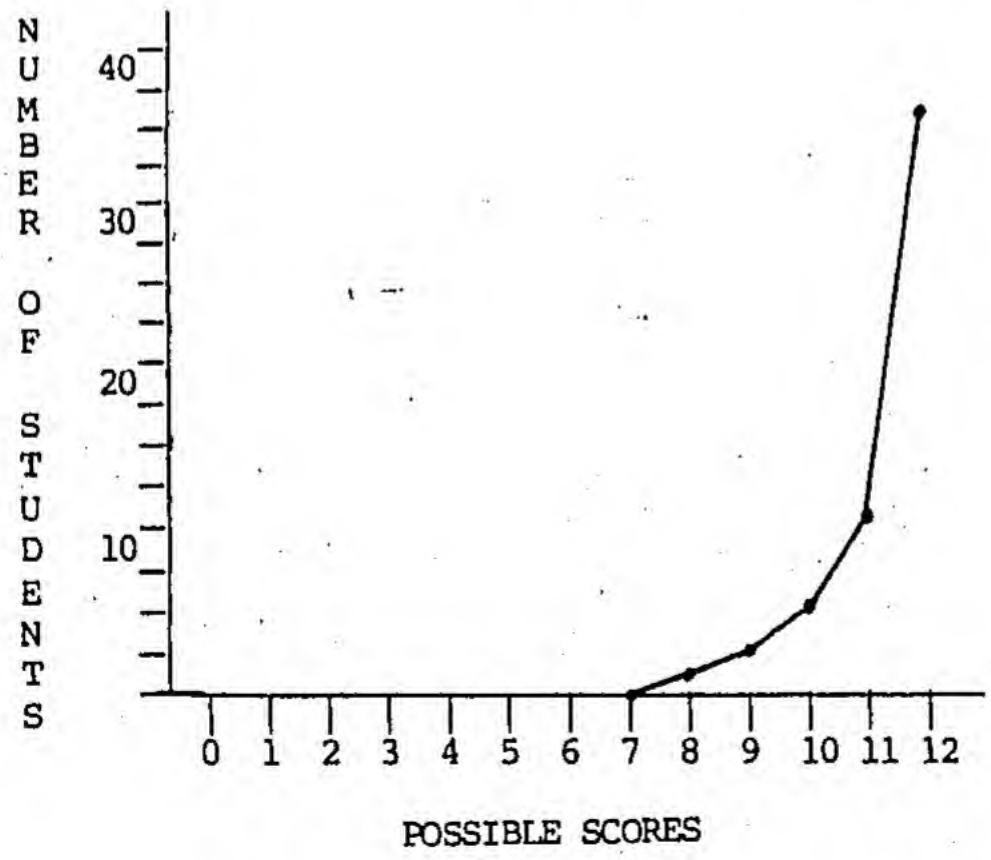
The only attitude scale showing a greater negative than positive distribution was the attitude toward editing a program. Almost three-quarters of the students (42/58) responded negatively with the greatest cluster of scores falling in the middle of the lower (negative) half of the scores. Children, while responding positively to all other measures of their attitudes towards computers and their work with them, felt negative about editing a program.

Attitude Toward Time Spent Working with Computers

All scores were in the 8-12 range (0 being most negative; 12 being most positive) with 48 out of 58 children scoring either 11 or 12. All children felt positive about their time spent working with computers. Figure 1 illustrates this positive trend.

Figure 1

Attitude Toward Time Spent Working with Computers

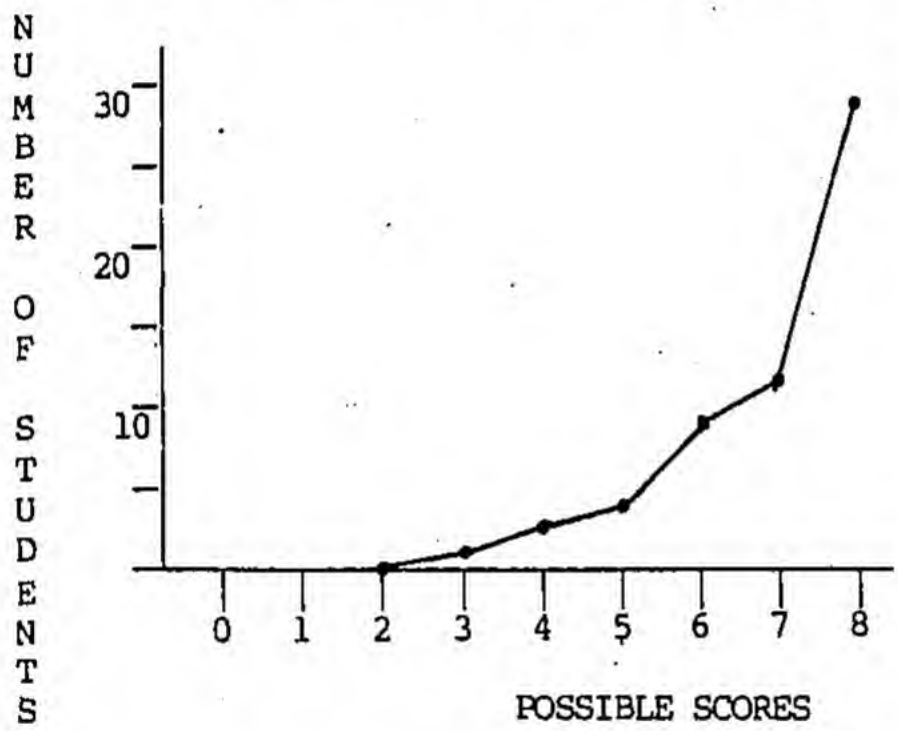


Attitude Toward Writing a Program

While all scores were in the upper two-thirds, only 8 out of 58 were in the middle third, leaving 50 out of 58 children with scores in the upper third of the scale (6,7, or 8). Most all children felt positive about writing programs. Figure 2 illustrates this positive trend.

Figure 2

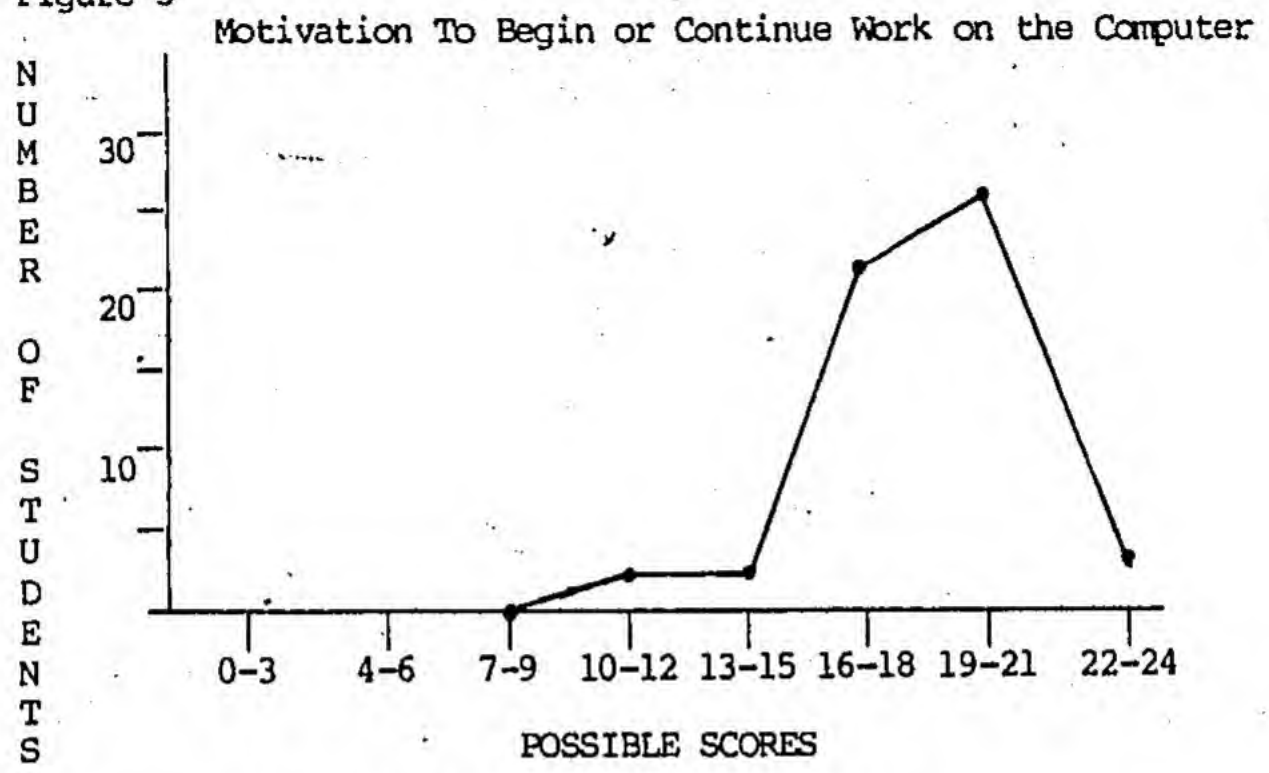
Attitude Toward Writing a Program



Motivation to Begin or Continue Work on the Computer

While all but two children scored in the upper half of the range of scores, most scores were clustered just before the highest scores. Almost all children felt positively motivated to begin or continue work on the computer. Figure 3 illustrates this positive clustering.

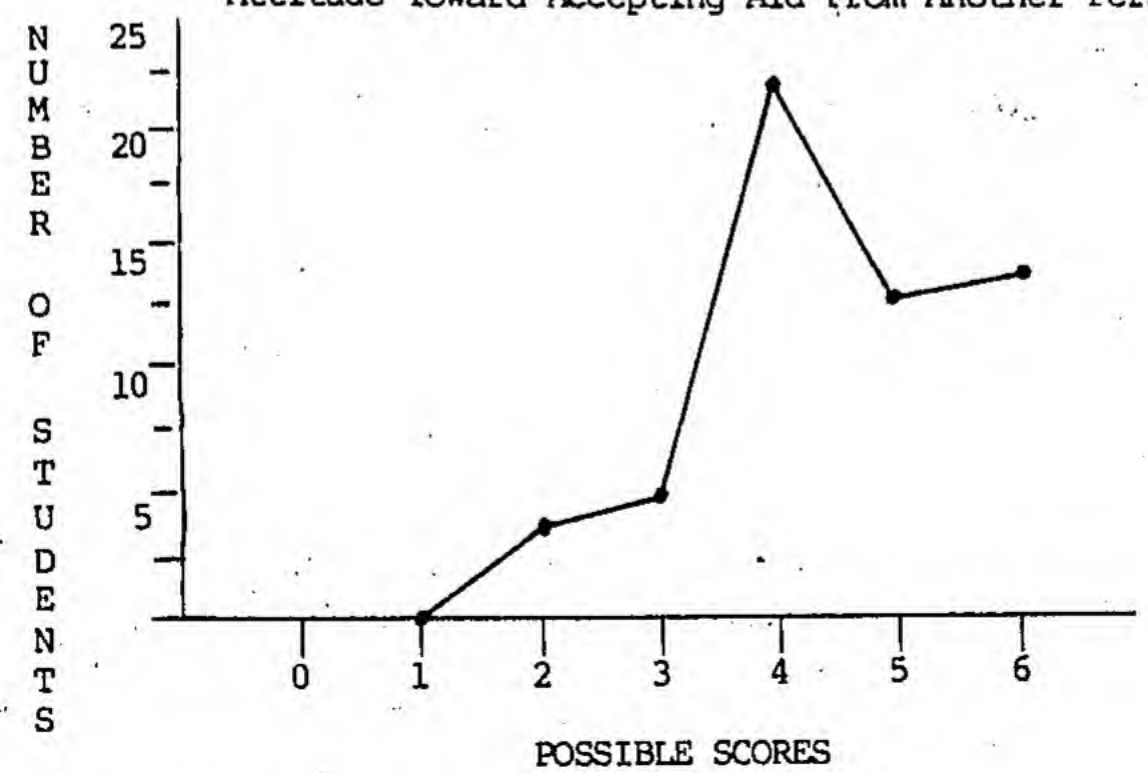
Figure 3



Attitude Toward Accepting Aid from Another Person

While 9 out of 58 scored at the midway point or just below, all others (49/58) felt positively about accepting aid. Most all children felt positive about accepting aid from another person while working at the computer. Figure 4 illustrates this positive clustering.

Figure 4

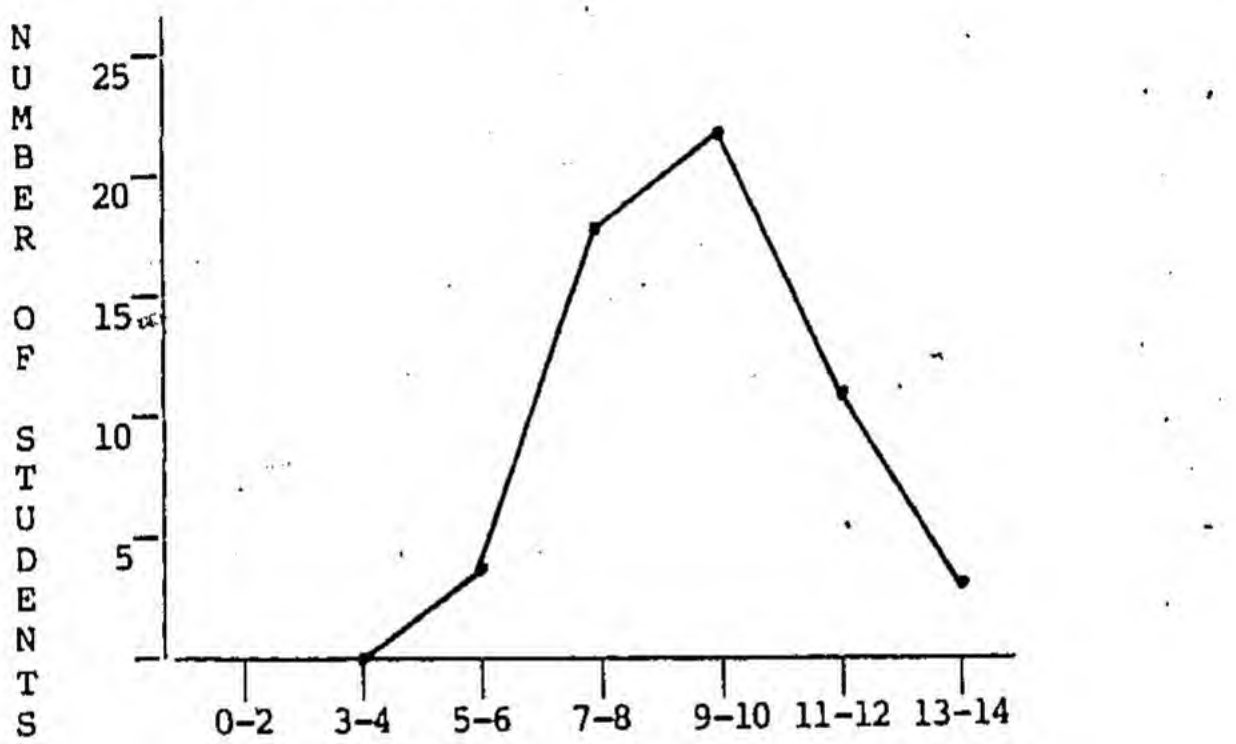


Self-confidence

All scores were in the upper two-thirds with scores clustering at about the two-thirds mark. Most children felt moderately self-confident with respect to their work with computers. Figure 5 shows this positive clustering.

Figure 5

Self-confidence

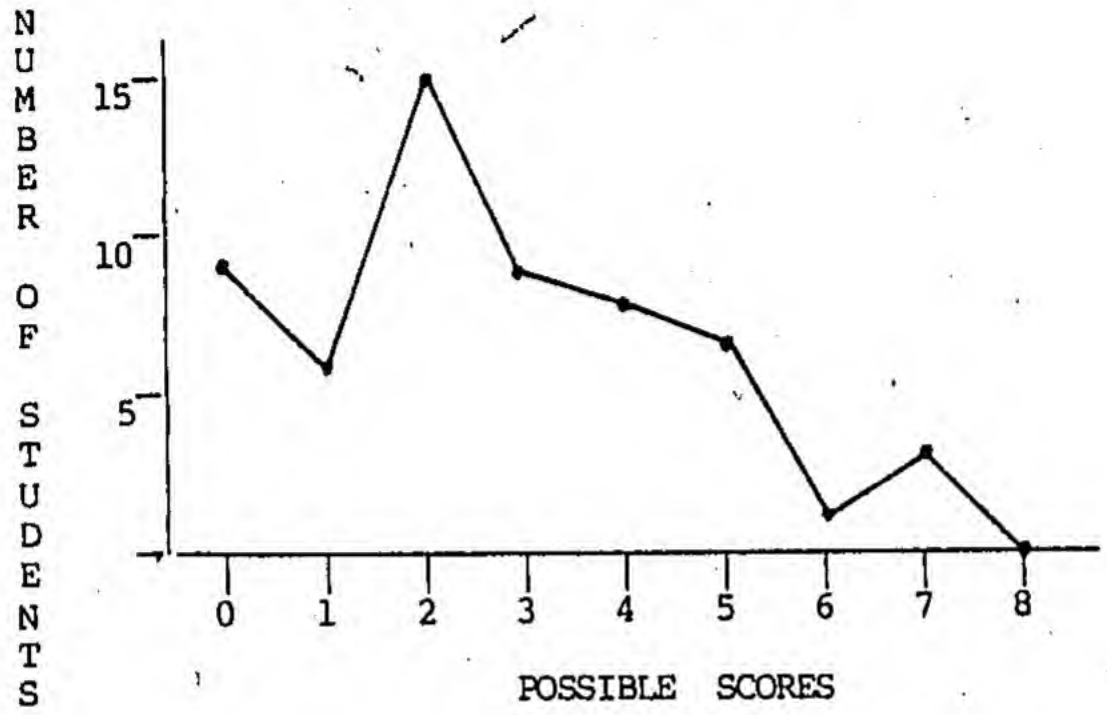


Attitude Towards Editing a Program

This was the only scale that indicated a negative attitude by the children in their work with computers. Approximately three-quarters of the students responded negatively towards editing their programs. Only four children scored in the upper third of the rating. The rest of the children, 54 out of 58, scored in the lower two thirds with a clustering at the one-third mark. Most children felt negative about editing a program. Figure 6 illustrates this negative cluster.

Figure 6

Attitude Towards Editing a Program



Comparison Between Groups

To what extent is programming ability influenced by sex, grade level, teacher, students' attitudes about themselves and their computer work, standardized reading and mathematics scores, amount of learning materials completed, or time spent working on the computer?

The Classification Schema used to categorize the students' original programs, written independently in the last laboratory session of the study, allowed for the division of all students into distinct groups according to programming ability. Two of the groups examined consisted of students who wrote high level programs (high level programmers) and students who wrote non-high level programs (non-high level programmers). High level programmers were those students whose original program was classified in the schema as a 5, a simple original program: a sequence of commands with purpose, a 6, a more complicated

original program, or a 7, an advanced original program. The programs written by students at this level displayed a greater sophistication with evidence of planning. Non-high level programmers were students whose original program was classified below five within the schema, thus anyone who was not categorized as a high level programmer.

It was expected that these two groups, high level programmers and non-high level programmers, would differ with respect to the amount of learning-materials completed. Table 4 shows the frequencies with which the high level programmers and the non-high level programmers completed their learning materials.

Table 4

Completion of Learning Materials Among Students Writing High Level Programs

	Programming Scores According to Schema	
Sections Completed	7, 6, 5	4, 3, 2, 1
1.1-1.3	15	33
2.1-3.2	7	3

The χ^2 test for two independent samples was used to determine the significance of differences between the groups in this category. Since it was expected that students who completed more sections in their learning materials would be, in fact, better programmers, the significance level was halved and a one-tail χ^2 test was used. For the data

in Table 4, the value of χ^2 is 3.76 which is significant at the 0.05 level. There is a significant learning materials completion difference among students who wrote high level programs. That is, children who wrote high level programs showed a greater frequency of completing more sections in their learning materials, or that those children who completed more sections in their learning materials were more likely to write high level programs.

It was also anticipated that high level programmers when compared to non-high level programmers would score higher, that is more positively, on the motivational scale of the attitude survey. Table 5 shows the frequencies with which the high level programmers and non-high level programmers are categorized by their motivational scores.

Table 5

Motivation to Begin or Continue Work on a Computer Among High Level Programmers

	Programming Scores According to Schema	
Motivation Scores	7, 6, 5	4, 3, 2, 1
9-16	1	10
17-24	21	26

The χ^2 test for two independent samples was used. Since it was expected that students who were more motivated to begin or continue work on the computer would be, in fact, better programmers, the signi-

ficance level was halved and a one-tail test was used. For the data in Table 5, the value of χ^2 is 3.40 which is significant at the 0.05 level. Although in general there were no low motivational scores, there was a significant motivational difference in the children who wrote high level programs. That is, children who wrote high level programs showed a greater frequency of high level motivation to begin or continue working on the computer, as compared to middle-level motivation, than children who did not write high level programs.

There was a noticeable difference in the level of programming ability in one of the third grade classrooms, Class C. Table 6 shows the distribution of high, middle, and low level programs in all four classrooms.

Table 6

Programming Levels Among Class Groups

	Programming Scores According to Schema		
Class	7, 6, 5	4, 3	2, 1
A	5	4	5
B	5	5	5
C	8	5	1
D	4	7	4

As can be seen from Table 6, Class C had more children with high-level programming scores and fewer children with low-level scores than any

other class in the study. The teacher in this class enjoyed the challenge of learning to program in PILOT herself and spent many extra hours working on her own projects. She also wanted to help those children who seemed reluctant to use the computer feel comfortable with the machine and programming. She would often plan a time to work with someone on the computer who needed a little encouragement. This extra attention in the beginning seemed to help give many of the children that needed confidence to work on their own.

Students who wrote high level programs were compared to students who wrote non-high level programs in relation to sex; other attitudes measured by the Attitude Survey; specifically, editing, self-confidence, accepting aid, time spent working with computers, and writing programs; and grade level. There were no significant differences in any of these categories for those two groups.

Students who wrote low level programs, a one or two in the Classification Schema, were compared to students who wrote non-low level programs. There were no significant differences in any category explored, namely amount of materials completed, sex, and attitudes measured in the Attitude Survey. The same was found in the sex difference category; that no significant differences were found in any category explored in relationship to sex differences.

The third grade students in the study sample were given a standardized reading and mathematics test (New York State Elementary School Test - Grade 3 - Form R) as a state requirement. There were no significant difference in the mathematics scores of high level programmers and non-high level programmers or of low level programmers and non-low

level programmers. The standardized reading scores of the third graders allows for the division of students into subgroups based on their placement within the group. Students who scored in the top two-thirds of the reading scores were compared to those scoring in the bottom third of the scores. These two groups were shown to differ with the respect to the level of programming ability, that is, those who wrote high level programs and those who wrote non-high level programs. Table 7 shows the frequencies with which the students who scored in the top two-thirds of the reading scores and those who scored in the bottom third of the reading scores are categorized by the level of programming ability.

Table 7

Reading Scores on New York State Elementary Reading Test Among Third Graders Writing High Level Programs

	Programming Scores According to Schema	
Reading Scores	7, 6, 5	4, 3, 2, 1
29-53	11	8
9-28	1	9

The χ^2 test for two independent samples was used. For the data in Table 7, the value of χ^2 is 4.38 which is significant at the 0.05 level. There is a significant reading test score difference among the third grade children who wrote high level programs and those who wrote non-high level programs. That is, children who wrote high level pro-

grams showed a greater frequency of high or middle level reading scores on a standardized reading test than children who did not write high level programs. The third graders who scored in the top two-thirds of these reading scores were more likely to write high level computer programs than children who scored in the bottom third of these reading scores.

While every attempt was made to keep the reading level of the learning materials low by using a limited vocabulary, children still needed to be able to read to work comfortably with the materials. All information was not transferred by reading, but a struggle with the reading may have been discouraging to some children.

For the last four weeks of the field trial, there was a group of 16 children who attended a computer afterschool program for an hour and a half each week. Many different computer activities were available during that period, including PILOT. Each of the 16 worked some of the time programming in PILOT so that on an average they accumulated fourteen hours of PILOT computer experiences by the end of the field trial compared to twelve hours for the other students in the sample. The participants in the afterschool program did not use their learning materials during that time but were able to write programs with occasional input from the computer afterschool instructor. Table 8 shows the frequencies with which afterschool computer participants and non-participants are categorized by the level of programming ability.

Table 8

Afterschool Computer Program Participants Programming Ability

	Programming Scores According to Schema	
Groups	7, 6, 5	4, 3, 2, 1
Afterschool Participants	11	5
Non-Participants	11	31

The χ^2 test for two independent samples was used. For the data in Table 8 the χ^2 value is 7.20 which is significant at the 0.01 level. There is a significant programming ability difference in children who attended the computer afterschool program. That is, children who participated in the computer afterschool program showed a greater frequency in the ability to write high level programs.

From teacher records and observations, children seemed to develop a greater sense of confidence in their ability to work with the computer and the materials. For most this came with the passage of time, somewhere after the first four weeks of the field trial. By the end of the field trial, most children had developed this sense of confidence and many were able to incorporate much of what was new into their programming knowledge more readily. The afterschool program occurred at a time when the ability to do this was high. An additional exposure

to PILOT at a time of increased confidence, seems to provide children with the ability to use, experiment with, and understand more commands and structures as reflected in their ability to write higher level programs.

Other Observations

According to the Questionnaire for Parents on Computers in the Home distributed in September 1982, it was shown that five families in the study sample had computers. Of those five, three had Atari computers but did not have the PILOT language cartridge. Midway through the study, one other family purchased an Atari computer. By December, three families had Atari PILOT available in the home.

The three boys who had increased accessibility to PILOT had a great advantage over the other children. One boy, whose father took an interest in PILOT and often worked with him, knew much more than the other children. A problem arose in this boy's third grade class, when he, joined by a few of his classmates, flaunted his expertise. Other children were very much intimidated by this and were refusing their time on the computer. A class meeting was held in which all children were able to voice their concerns and to realize that they had to accept where they were in their own PILOT work as well as respect the work and attempts of their classmates. The situation improved; seldom did anyone refuse to work on the computer again in this class.

Most students were not used to working in workbooks except in reading. The format of the learning materials varied from these more traditional workbooks in that the presentation of material was

continuous rather than being a series of related, but discrete exercises. Many of the initial difficulties with working in the materials seemed to be connected with this new form of written material. After some time, the length of which depended on the individual, the child came to see that he/she could be independent in the use of these materials; that with this booklet and some reading and exploring, he/she understood much about programming.

While the materials provided children with the vehicle for learning to program, it was found through interaction with the children and suggestions from the teachers, that certain revisions in the materials would improve both the content and style. Often the children were watching the keyboard and missed what had happened very quickly on the screen. For this reason, children need to be alerted through the materials when there was something very important about to happen on the screen that needs their undivided attention. Some symbol or phrase, such as "Watch Carefully" needs to be added in those places. It was found at the end of the first section that more exercises to provide practice with the commands, movement of the turtle, and sequentiality would have been helpful. Adding color earlier would give the children opportunities for experimenting with order as they might want to make the triangle red and would have to decide when the turtle needed to pick up his red pen and when he needed to draw the triangle. Explorations are especially fruitful; more should have been provided within the materials, especially in the beginning. The materials have been revised with these ideas in mind. A copy may be obtained from the author of this paper.

Programming in PILOT provided the children with experiences in decision-making, cooperation, and problem solving. Cooperation was often necessary when two children shared one computer. Children often solved one problem by use of a related one or through mathematical relationships seen or explored, as in the polygon microworld exercises. Children used their body or their ability to form mental images to solve a turtle geometry problem. Adding and deleting lines in a program gave the children a sense of order within a program, extending their known idea of sequentiality in other areas to programming.

One drawback in the use of PILOT was the absence of a visible turtle. In order to see how much the "turtle" had turned, the child needed to draw a line immediately after the turn. In the Logo programming language, the turtle is visible and turns when directed to do so in the direction and quantity asked for.

Chapter VI

SUMMARY AND RECOMMENDATIONS

Summary

The learning materials for the study reported here were developed as a vehicle through which second and third graders could learn the turtle graphics computer language Atari PILOT in both a laboratory setting where expert assistance was available and in a classroom setting where limited assistance was available. The study was designed to look at the self-pacing/individualizing capabilities of the learning materials. An attempt was made to keep the reading level low; important programming commands, structures, and/or ideas were presented whenever possible by computer observation, illustrations, screen displays, and simple phrases and sentences. While children would all follow the same sequence of activities in their learning materials, individual explorations would be encouraged.

The materials were designed to be written in by the user. In contrast to the format of a workbook, the presentation of material was continuous rather than a series of discrete exercises, with knowledge and understanding building on that of the previous activities. Observations were to be made and recorded, questions answered, and programs completed, edited, or modified. It was intended that the materials would provide a way of introducing the programming language PILOT to the students so that they would use the information obtained in this way to explore independently of the materials. In this way the

materials would provide a model for programming.

Instruments were developed to help evaluate the research questions of the study which explored the achievement level, extent of self-pacing achieved, and assistance given in the use of the learning materials; the level of programming ability at the conclusion of the field trial; student attitudes; and the extent to which programming ability was affected by other factors. Instruments include tests, information recording forms, a program classification schema, and an attitude survey; all instruments but the standardized reading and mathematics scores were developed specifically for this study.

The field trial site was a New York City private school with a student population from a variety of academic levels; the entire second and third grade classes used the materials over a twelve week period. Prior to the field trial, teachers worked out scheduling problems to assure for equal access to the computers in their classrooms; some preprogramming work was done with both the teachers and the children. The learning materials were introduced in the laboratory setting; subsequent PILOT programming activities were done during both the students' weekly laboratory period and class time. It was expected that students work in their materials and/or explore. Multiple-choice Computer and Programming Tests were administered upon completion of Part I, II, and III. Records were kept by teachers and the laboratory instructor on assistance given, completion of tests and materials, and problems. During the final week of the field trial, each child was asked to write a program of his/her own under supervision in the laboratory; the Attitude Survey was administered. A self-selected

group of children participated in a computer afterschool program for an additional two hours of PILOT programming experience near the end of the field trial period.

Findings

Children were able to learn the programming language, Atari PILOT, using the self-paced learning materials designed for that purpose. While assistance of many kinds was needed throughout the students' work in the materials, the group was able to use the materials fairly independently by the fourth week of the study to learn about, and better understand PILOT commands and structures. By the end of the twelve weeks of the field trial, all of the children (58/58) had completed the first two sections of the materials, 47% (28/58) the third section, 17% (10/58) the fourth section, 5% (3/58) the fifth section and 3% (2/58) the sixth and seventh section, indicating that to a rather high extent, self-pacing was realized in the use of these learning materials.

In general, children who completed more than two sections in the learning materials completed them more accurately. Incorrect answers in the first two sections were often just blanks, many of those asking the child to draw conclusions based on computer observations. The items left blank were often those that involved writing down more than one word or number, and from observation, may well be the reason why those items were left blank. A few children also exhibited some difficulty in editing a program to achieve a given effect. This may be due, in part, to the general negative attitude towards editing a program as shown in the Attitude Survey. Class differences in comple-

tion of materials were significant. All students who took Multiple-choice Computer Tests and Programming Tests, designed to check on the understandings in each part of the learning materials, were able to do so at the mastery level or better.

Children did need assistance when working on the materials as documented through teacher observation forms and anecdotal records of laboratory observations. There was a greater dependence on others in the beginning of the field trial. As students' confidence in their ability to control the computer, write programs, and work through the materials grew, less assistance was needed. The kind of assistance also changed. Many children came to rely more on their peers where that was allowed. Teachers and others were giving help in program writing, editing, and problem solving more than in computer related and/or reading areas. While it was intended that the materials be used independently, assistance was not withheld; in fact, often questions were asked of the students to help them better understand some idea or move away from their materials, upon which some children had become dependent, to explore what they had taken in thus far.

In writing sample programs at the end of the field trial, it was evident that while all children could write computer programs, the range of programming abilities was quite extensive. The sample programs were rated according to the level of sophistication based on the Classification Schema developed for that purpose, ranging from a simple copied program to a program with looping and counting structures built in. Fifteen children wrote low-level programs, sequences of random commands or ones copied from another source. These children had

some ability to use commands, but showed no evidence of editing or a sense of order. Twenty-one students wrote middle level programs, explorations with attempts to use commands and structures. While there is evidence of editing abilities, there is an absence of planning in the explorations. Twenty-two students wrote high level programs, planned original programs often including a more sophisticated structure. There is evidence of planning and some understanding of the structures used. The three levels of programming, among which the sample was almost equally divided, may be given as imitative, exploratory, and planned.

The three levels of programming seen are similar to those defined by Solomon (1982) in the Brookline Project and Howe (1980) in the Edinburgh Project in working with children during their first year of learning to program. The study presented here defines programming ability by the programs written. At the low level, imitative, there is little evidence of planning or editing. Howe sees this first stage, through programs written, as product-oriented, drawings being produced without much care given to the process. Solomon, in looking at the style of the learner, gives the first stage as microexplorer, the learner needing time to explore small segments of the language. Small explorations, often imitative in nature, are common elements of this level of programming in the studies mentioned. At the middle level in this study, explorations are still evident but now are seen coupled with the ability to edit programs to achieve certain effects with no evidence of planning. Howe sees this stage as style conscious where changes are made based on what the learner perceives as being good

form; the absence of planning is seen. Solomon describes this middle level programmer as the macro-explorer, one who experiments to achieve a final product without beginning with a set plan. Greater explorations without initial planning is the common element in this level of programming for the three studies. The high level program, as detailed in the study reported here, is one which shows evidence of planning and the understanding of some structures. Howe describes this stage as the problem solving stage, one in which the programmer uses available resources to solve the problem at hand. Solomon calls the programmer at this stage the planner. The presence of planning and the attempt to follow through on a plan are common elements in programming at this level in the three studies mentioned. Similar levels of programming or programmer style were seen in the three studies despite the fact that the Howe and Solomon studies dealt with programming in Logo and this study dealt with programming in PILOT.

It has been found in this study that at a significant level, high level programmers completed more of their learning materials than non-high level programmers. That is, students who complete more of the learning materials are more likely to write a high level program or, high level programmers are more likely to have completed more of their learning materials. While all students were found to be highly motivated to begin or continue work on the computer, students who wrote high level programs have been found to be more highly motivated at a significant level. Noticeable but not significant differences were found in the programming levels of one class. This particular class had fewer students who wrote low level programs and a greater number of

children who wrote higher level programs than any of the other three classes. It was felt that the high degree of assistance given to those students by their teacher in the beginning stages of their work in the materials may well account for the difference.

The level of programming ability was significantly affected by factors that influenced equal accessibility. The two children whose programs were rated seven in the Classification Schema were two of three children who had Atari PILOT available at home. These children consequently spent more than twelve hours of time at the computer programming in PILOT. There was a significant difference in the ability to write high level programs in favor of the participants in an afterschool computer program as compared to non-participants. The difference may be due to having additional PILOT computer experiences at a time when most children had grown to a point in their computer and programming confidence where much more information was able to be assimilated. The availability of an expert during those extra hours spent working on the computer in PILOT may also have contributed to the high level of programming ability. These students did not show a statistically significant motivational difference when compared to those children who did not attend the computer afterschool program although none of the afterschool participants scored less than 17 on the motivation scale in the Attitude Survey as compared to 15 of the non-participants in that category.

Individual teachers were found to have an effect on the amount of learning materials completed. At a significant level, one second grade class completed less of their learning materials than the other three

classes as well as the other second grade. The teacher in the low-materials completion class was very apprehensive about the computer in general and felt that, as a new teacher, she had to give most of her time and energy to dealing with other areas of the curriculum. The other second grade teacher had a great deal of personal interest in the computer and gave much of his energy to understanding PILOT and working with and helping his children in their understanding. Teacher differences in computer interest and amount of time willing to devote to computers seemed to be a factor in determining the amount of learning materials completed at the second grade level.

Third graders scoring in the top two-thirds of the scores in a standardized reading test were shown to differ significantly from those students scoring in the bottom third of the scores in their ability to write high level programs. That is, children scoring in the top two-thirds on the reading test were more likely to write high level programs. While attempts were made to keep the reading level low in the learning materials, it remains that reading was still necessary in order to complete the materials. This may account for the differences in reading ability and programming ability since it has already been shown that amount of learning materials completed significantly effects the degree of programming ability.

Students attitudes towards their work with computers were measured by the Attitude Survey created for that purpose. As a group, students exhibited positive attitudes in all areas measured except editing a program. Most of the children felt extremely positive about their time spent working with computers and writing a program. Most children

felt very motivated to begin or continue work on the computer and very accepting of aid from another person. Children were, for the most part, self-confident about their work with computers at a moderate level. Most children showed a negative attitude towards editing a program. Motivation to begin or continue work on the computer was shown to be significantly related to the level of programming ability as displayed on the sample programs. That is, those students showing a higher level of motivation were more likely to write high level programs. No other correlations between attitudes or attitudes and other factors were seen.

Implications for Learning

The learning materials created for this study enabled children to learn the programming language Atari PILOT at their own pace with periodic assistance as needed, in an environment that included one hour of computer use each week, and interaction with a laboratory instructor, teachers, and peers. The materials were designed to provide children with an entry into programming experiences, not as a total program to be rigidly adhered to. Programming can be learned in the classroom with materials of this sort with a realization that other elements may be crucial in the learning of programming. Assistance is especially crucial in the beginning of the students' work with programming to help the child feel secure in his/her ability to work with the computer and programming. The expertise offered, in this case by a computer laboratory instructor, may be essential to the success of such a program built around a particular set of materials.

While many program types were seen in the sample programs written by the children at the conclusion of the field trial, much can be learned about programming ability by observation, too. Program types were based on evaluation of the children's sample programs according to the Classification Schema developed for that purpose. The observation of the children during programming yielded some insights into their programming ability that could not have been seen by the examination of a classification score alone. It was obvious, in one case, that a child made a change in his program that lowered his final evaluation of his program. It was also seen that children were using commands and structures that they had learned through interaction with other children. Observation provides us with greater insight into students' programming thoughts, development, understandings, and misunderstandings, that is not possible through a program sample or the classification of it. Classroom and laboratory observations written down made teachers especially aware of difficulties in learning to program.

Completion of learning materials, motivational attitudes, and degree of assistance in the beginning of their work in the materials were shown to effect the programming ability as exhibited in the sample programs. The more materials completed by a student, the higher the motivation to begin or continue work on the computer, and the more help received during the computer "reluctance" period, the greater the possibility that the student would write a higher level program. Interaction at times when something new is introduced, especially noticeable at the beginning of the work on the learning materials when

everything was new, seems to establish a sense of confidence in one's ability to make things happen at the computer.

As a group, students exhibited positive attitudes towards their work with computers except in the area of editing a program. Editing is done, in one instance, to correct some mistake in the program. A mistake needs to be viewed as a challenge so that working critically to make changes becomes a greater and more rewarding part of the child's learning experiences.

According to the standardized reading scores of third graders, the children who scored in the upper two-thirds of all scores were more likely to have written higher level programs. In order that reading ability not be a determining factor in the completion and understanding of materials, much attention must be given to keeping the reading level low. It may be the case that achievement levels affect the level of programming ability due to some underlying cognitive abilities, but that is not shown in this study. It can only be stated that the aforementioned difference was noted and that it was, in fact, necessary for children to be able to read to make full use of the materials.

Equal access seems to hold special significance in programming ability. All students were to have had approximately twelve hours of computer use during the field trial. The availability of PILOT in three homes and in the computer afterschool program increased the amount of computer time by two hours for 17 of the 58 students in the study. The result of this disruption of the equal-access situation is significant and especially worthy of note. There was a significant difference in the ability to write high level programs in favor of the

participants in the afterschool program. Most all children at this time in the study had grown significantly in their confidence to produce some interesting programs on the computer. When motivation and confidence levels are high, amount of learning may also be high. Two issues are seen here: equal access to computers and providing learning opportunities at a critical time for maximum learning. Both issues may be crucial to the amount of programming learned.

Implications for the Classroom

Teacher observations in this study were made to note interactions of the students with the learning materials. These classroom and laboratory observations were also helpful in studying how children learn something new. Teachers were forced to look at their interactions with children, trying to ascertain when to offer assistance, when to ask a question, and when to send them back to their own resources. Learning to program may be analagous to writing a creative story, with the need for teacher intervention at appropriate times. Observation provided the teacher with some information on individual understandings and misunderstandings; what the child knew.

Working with children on their programming work and/or materials at certain times in order to give them the needed confidence to work on their own has been shown to effect the level of their programming ability. This is especially crucial in the beginning of their work, when the confidence level is low due to the newness of the computer, programming and the materials.

Individual teachers affected the quantity and quality of work

completed in the learning materials. Assistance given to children having difficulty at crucial times helped children complete more of their learning materials. In one class, where the amount of materials completed was significantly less than in all other classes, the number of incorrect responses was also higher. The teacher in this class felt apprehensive about her ability to work with the computer, and that her time should be spent in other, more important areas. It may be crucial for the classroom teacher to have a certain commitment to programming in order to be assured that her students will grow in their understanding, especially when there is no other contact with computers and/or programming. Such was not the case here, since the children also received one thirty minute session per week in the computer laboratory.

These materials and materials of this type need to be viewed as a springboard into programming. They provide children with an introduction to programming and a form of expertise in the absence of a programming-knowledgeable teacher. It is expected that children venture forth from these materials since the exploration of commands and structures in a programming language will provide a clearer understanding of the power of those aspects of the language. The same rationale was a part of the Brookline Project. Watt (1982) found that the presence of the expert was indeed critical and that when this element was to be absent from the project the second year of their study, materials were developed that would attempt to replace the expert and help children learn programming in a classroom setting with teachers just beginning their own work with Logo. These materials were

also designed to provide the children with an entry into programming; leaving the materials to work on projects of one's own choosing was expected.

Equal computer access seems especially important. Those who participated in the afterschool computer program had approximately two extra hours of computer time, rating higher in their programming ability. Their work with PILOT during those extra hours may have been enhanced by the presence of an expert. A great effort was made to provide for equal access in each classroom so that all would be assured a time on the computer without pressure to give up one's time. The afterschool computer program seems to provide support to the importance of that issue.

Recommendations

The equal access/unequal access question raised here suggests the need for further research. The level of programming ability was significantly affected by the amount of access to PILOT in favor of the group with the greatest amount of time spent on the computers (approximately 14 hours as compared to 12 hours). It should be determined whether greater accessibility influences programming ability as well as the conditions necessary for this to occur. For example, it should be determined whether the presence of an expert during the time of increased computer use is a necessary factor for the imbalance of programming ability to occur. The motivational factor and its influence on programming ability and greater computer access should be studied. It was speculated that the differences in programming ability

due to greater computer access was a result of having additional computer experiences at a time when most children had grown to a point in their computer programming confidence where much more information was able to be assimilated. This hypothesis is worthy of further study. Perhaps greater access at any time during the programming period would have yielded the same results. Providing that extra access at the beginning of the work in a programming language may supply some interesting information for study.

While looking at the categories of the programs that children write may be helpful in studying the levels of programming, finding effective ways of assisting the child in his/her move to being a more effective programmer and thus capable of writing higher level programs may be worthy of study. Finding the component skills of programming may lend more insight into this search. More information on prerequisite skills needed to move from one level of programming ability to another may assist in a study of these levels.

References

Agenda for Action: Recommendations for School Mathematics of the 1980's. Reston, VA: National Council of Teachers of Mathematics, 1980, 8-11.

Howe, J. A. M. Developmental Stages in Learning to Program. In F. Klix & J. Hoffman (Eds.), Cognition and Memory: Interdisciplinary Research of Human Memory Activities. Amsterdam: North-Holland, 1980.

Introductory Turtle Geometry Student Activities Book. Brookline, MA: Brookline Public Schools, 1981.

LOGO II: A Curriculum for Intermediate Grade Students. Boston, MA: The Education Collaborative, 1982.

Luehrmann, A. Computer Literacy - What Should It Be?. The Mathematics Teacher, 1981, 74 (9), 682-686.

Papert, S. Computer-Based Microworlds as Incubators for Powerful Ideas. Cambridge, MA: MIT Artificial Intelligence Laboratory, 1978.

Papert, S. Mindstorms. New York: Basic Books, 1980a.

Papert, S. New Cultures from New Technologies. Byte, 1980b, 5(9), 112-115.

Papert, S. Teaching Children Thinking (Logo Memo Number 2). Cambridge, MA: MIT Artificial Intelligence Laboratory, 1971a.

Papert, S. Teaching Children to be Mathematicians vs. Teaching About Mathematics (Logo Memo Number 4). Cambridge, MA: MIT Artificial Intelligence Laboratory, 1971b.

Papert, S., Watt, D., diSessa, A., & Weir, S. Final Report of the Brookline Logo Project: An Assessment and Documentation of a Children's Computer Laboratory. Cambridge, MA: MIT Division for Study and Research in Education, 1979.

Pea, R. D. Logo Programming and Problem Solving (Technical Report Number 12). New York: Center for Children and Technology, Bank Street College, 1983.

Pea, R. D. & Kurland, D. M. On the Cognitive Effects of Learning Computer Programming: A Critical Look (Technical Report Number 9). New York: Center for Children and Technology, Bank Street College, 1984.

Solomon, C. A Case Study of a Young Child Doing Turtle Graphics in Logo (Logo Memo Number 28). Cambridge, MA: MIT Artificial Intelligence Laboratory, 1976.

Solomon, C. Introducing Logo to Children. BYTE, 1982, 7 (8), 196-208.

Student PILOT Reference Guide. Sunnyvale, CA: Atari, Incorporated, 1981.

TI Curriculum Guide. Lubbock, TX: TI, Incorporated, 1982.

Watt, D. A Comparison of the Problem Solving Styles of Two Students Learning Logo: a Computer Language for Children. Proceedings of the National Educational Computing Conference, 1979a, 255-260.

Watt, D. Final Report of the Brookline Logo Project: Profiles of Individual Student's Work. Cambridge, MA: MIT Division for Study and Research in Education, 1979b.

Watt, D. Logo in the Schools. Byte, 1982, 7 (8), 116-134.

Appendix A: Copy of Learning Materials

PILOT

A PROGRAMMING LANGUAGE FOR KIDS . . . AND THE ADULTS IN THEIR LIVES

Programming in PILOT

1. When you see words and numbers printed like this, read them.

   Type this into the computer.
   Run the program.
   Change line 30.

2. When you see words and numbers printed like this, type them.

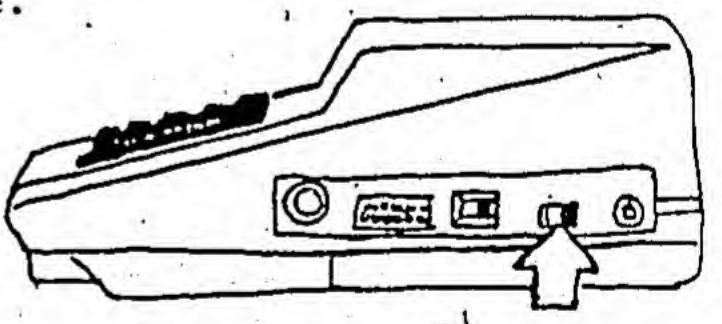
```
10 GR:DRAW 15
RUN
GR:CLEAR
```

3. Press the return key after every line you type.

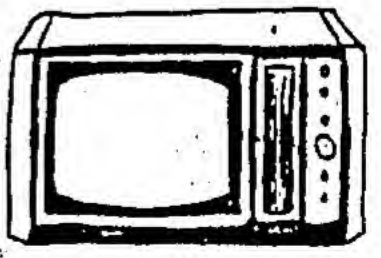
```
RUN (RETURN)
60 GR:TURN 120 (RETURN)
```

4. Follow these steps to get the computer ready.

   1. Put in the PILOT cartridge.



   2. Turn on the computer.



   3. Turn on the television.

Programming in PILOT: Part 1

Section 1: Some turtle graphics commands

GR:DRAW

GR:TURN

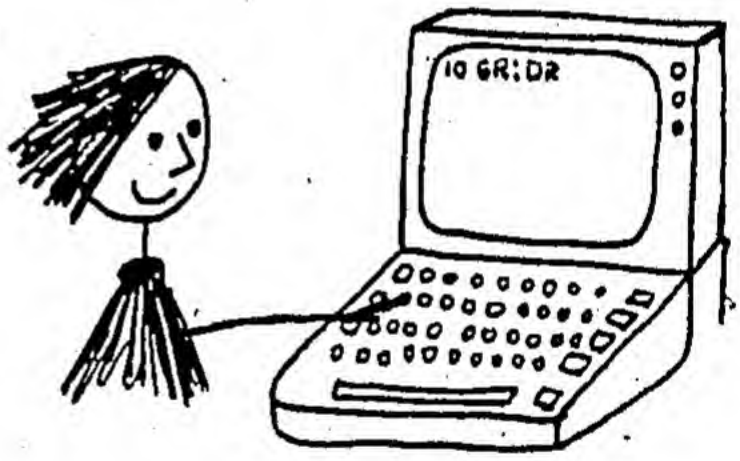
GR:CLEAR

GR:QUIT

GR: means graphics

Type this into the computer.

```
10 GR:DRAW 15
20 GR:TURN 120
30 GR:DRAW 15
40 GR:TURN 120
50 GR:DRAW 15
60 GR:TURN 120
```



RUN

GR:CLEAR

Think!

What happened?

RUN

LIST

Think!

Where did the lines of the program go?

GR:QUIT

LIST

Run the program. Draw a picture of the run. ⟶	

GR:CLEAR

GR:DRAW 25

RUN GR:DRAW 25 Draw the picture you see. ⟶	

GR:CLEAR

Think!

RUN

Where did the extra line go?

List the program.

60 (RETURN)

LIST

What is missing from this program? _____

RUN

GR:DRAW 25

Draw the picture you see. ⟶

60 GR:TURN 120

List the program.

Take out another line from the program.

Which line did you take out?_____

RUN

Draw a picture of the run. ⟶

Type that line again into the computer.

RUN

GR:QUIT

LIST

Take out another line from the program. Which line did you take out? _____ RUN Draw a picture of the run. ⟶	

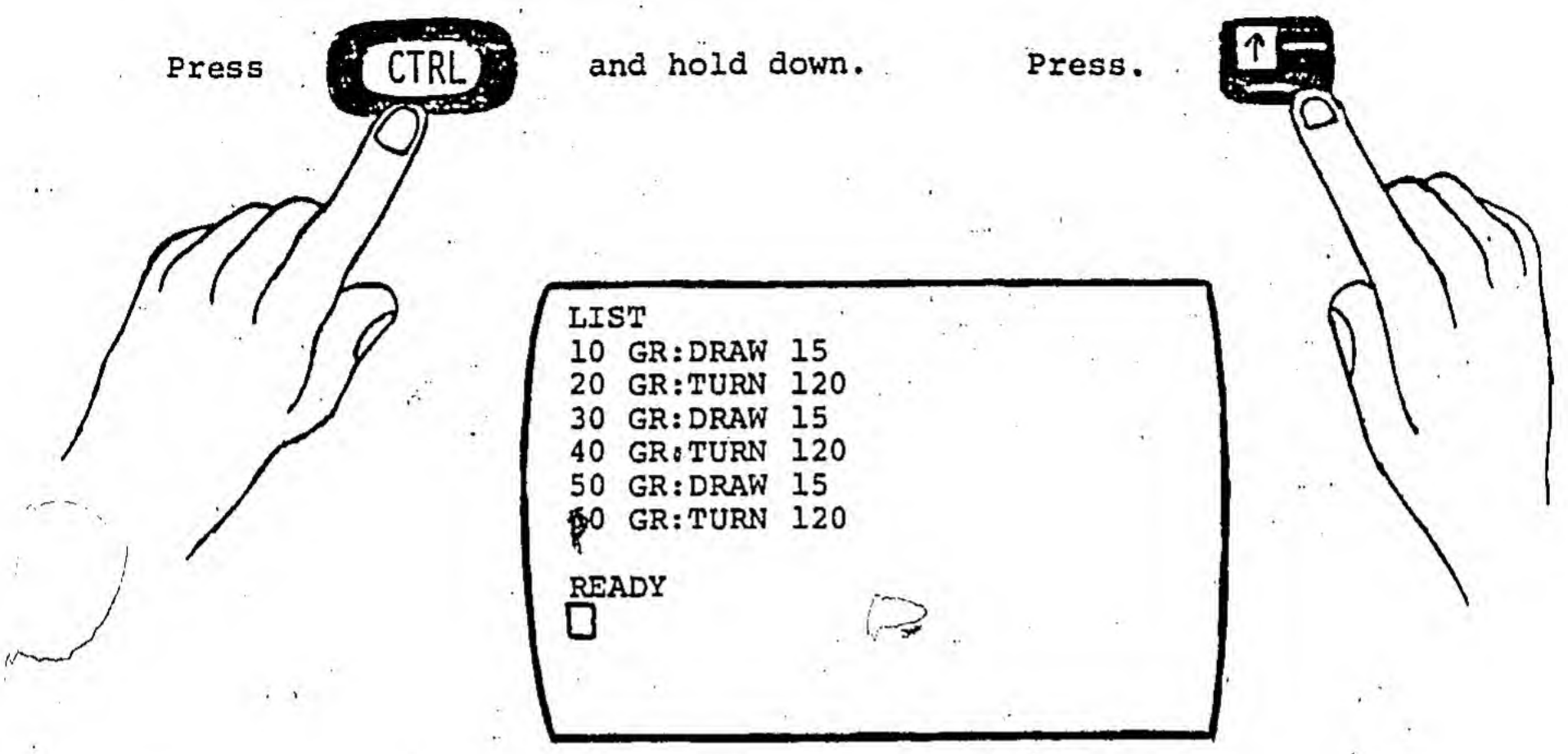
GR:QUIT

LIST

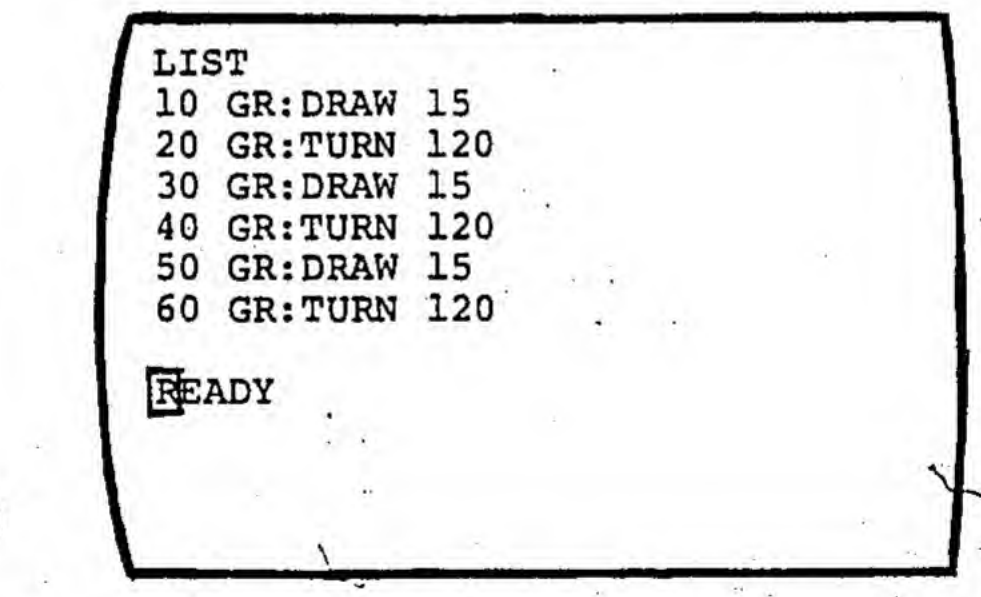
Fix your program so that it has all the line numbers again.

Does it look like this?

```
10 GR:DRAW 15
20 GR:TURN 120
30 GR:DRAW 15
40 GR:TURN 120
50 GR:DRAW 15
60 GR:TURN 120
```



The cursor moves up one line.



Move the cursor again.

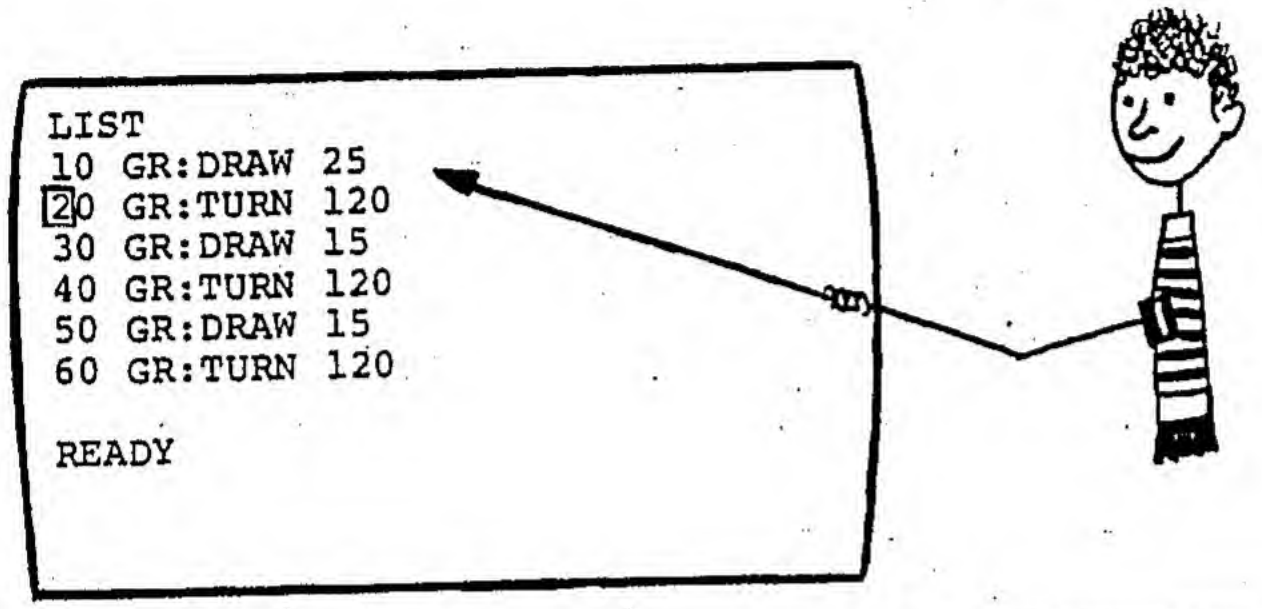
Move the cursor until it is like this. 10

Move the cursor this way ⟶ until it is like this. 15

Press (2) (RETURN)

Think!

What has changed?



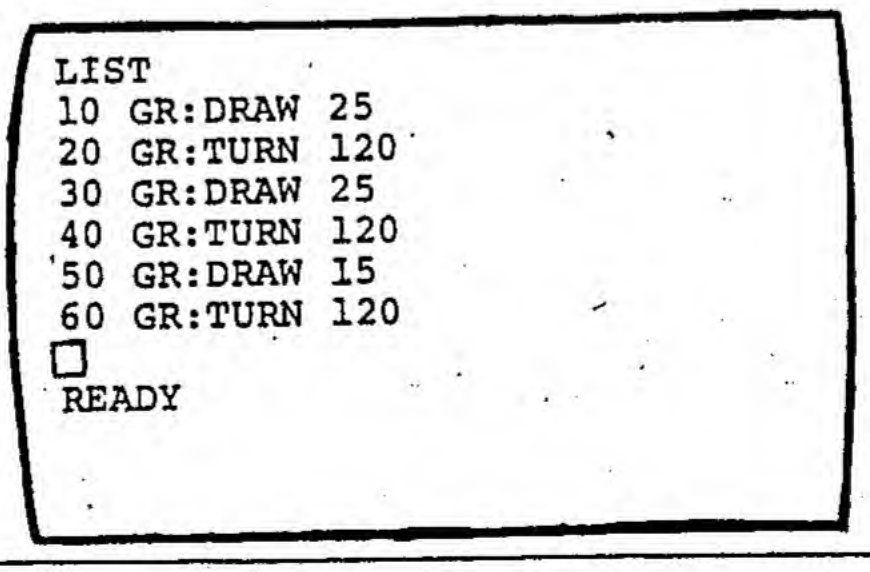
Press (RETURN)

Now change line 30 to

30 GR:DRAW 25 (RETURN)

Press (RETURN) (RETURN) (RETURN)

Does your computer screen look like the one below?



RUN

Think!

What happened?

GR:QUIT

LIST

Type this line.

50 GR:DRAW 25

LIST

What happened to line 50? _____

RUN

Exercises

List the program.
It should now look like this.

```
10 GR:DRAW 25
20 GR:TURN 120
30 GR:DRAW 25
40 GR:TURN 120
50 GR:DRAW 25
60 GR:TURN 120
```

1. Make the triangle bigger.

LIST RUN

2. Make the triangle smaller.

LIST RUN

3. Try a different turn number.

LIST

RUN

4. Change the program in any way you want.
Here is one done for you.

LIST

```
10 GR:DRAW 40
20 GR:TURN 120
30 GR:DRAW 15
40 GR:TURN 120
50 GR:DRAW 15
60 GR:TURN 120
```

RUN

Now you make a change in the program.

LIST

RUN

NEW AUTO

NEW

```
10 GR:DRAW 35
```

RUN

GR:QUIT

LIST

NEW

LIST

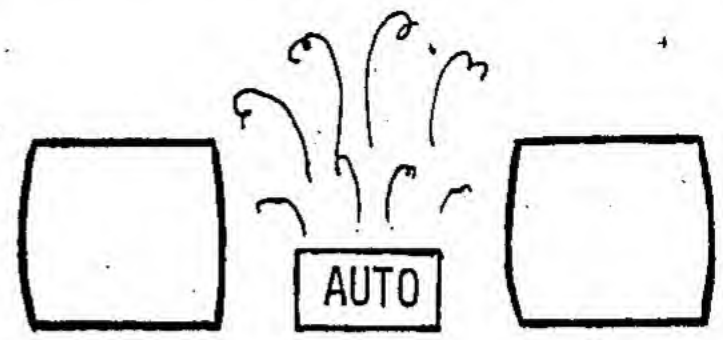
Think!

What happened to your program?

The NEW command erases the program from the computer's memory. The computer is then ready for a new program.

Now for a little COMPUTER MAGIC.

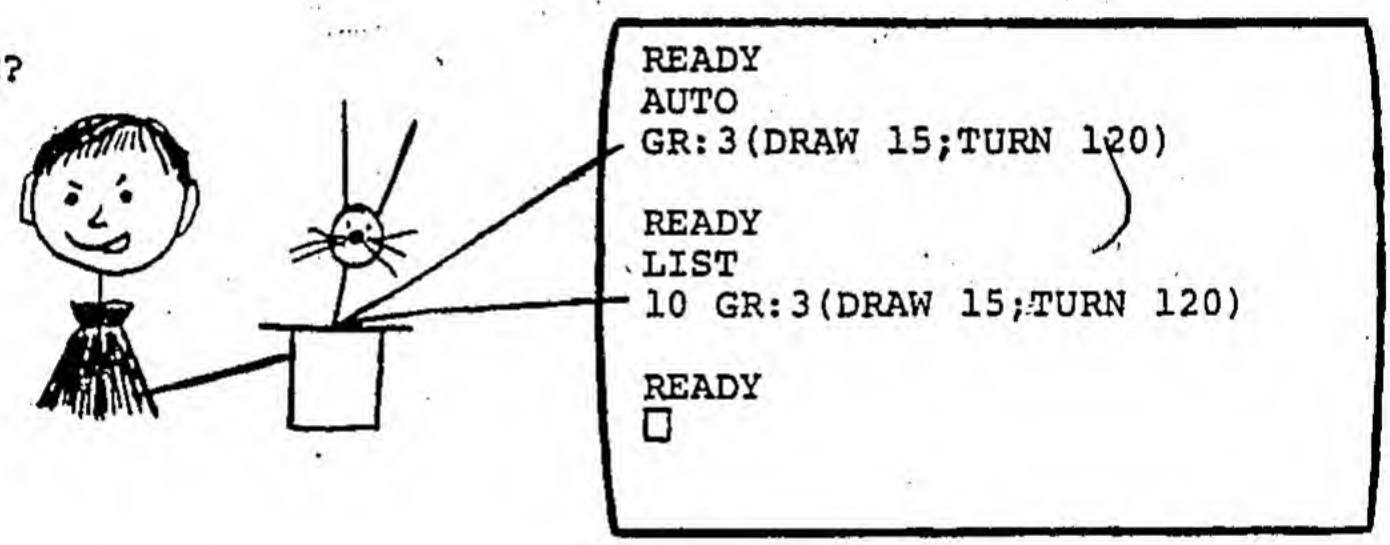
AUTO



GR:3(DRAW 15;TURN 120) RETURN RETURN

LIST

More MAGIC?



The AUTO command lets you enter a program into the computer. The computer puts the line numbers on for you.

RUN

Still more MAGIC?

This program

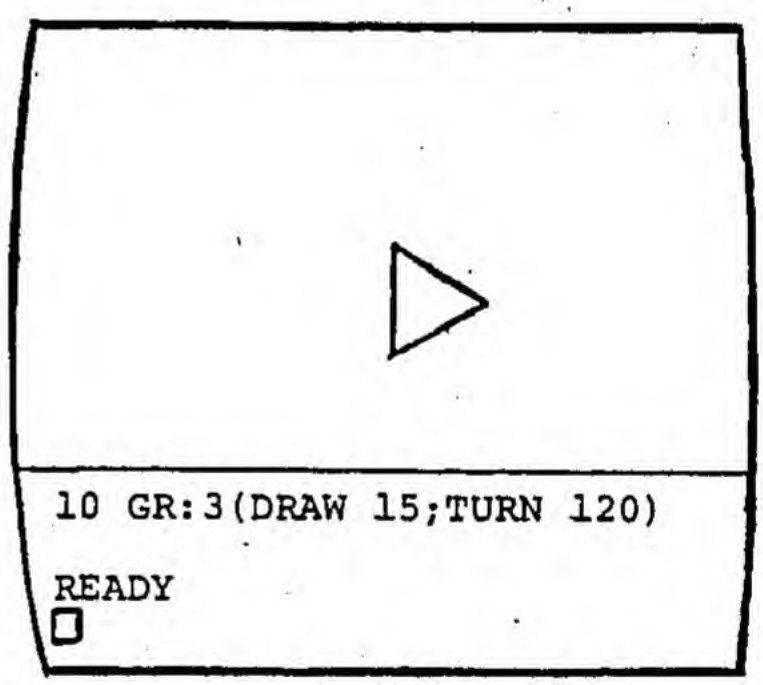
```
10 GR:3(DRAW 15;TURN 120)
```

draws the same picture as this program.

```
10 GR:DRAW 15
20 GR:TURN 120
30 GR:DRAW 15
40 GR:TURN 120
50 GR:DRAW 15
60 GR:TURN 120
```

Do not type GR:QUIT.

LIST



Move the cursor to the 15 in line 10.

Change the number to 30. RETURN

RUN

In these exercises you will make some interesting programs.

This program makes a triangle. (3 sides)

```
10 GR:3(DRAW 15;TURN 120)
```

1. Change this program to make a pentagon. (5 sides)

   The 3 should be changed to a _____.
   Make that change in your program.

   The TURN 120 should be changed to TURN _____.
   Try a number you think is right.

   RUN Think!

   Do you need a larger number or a smaller number now?

   Keep trying until you find the right number.

   LIST RUN

   10 GR:_____

2. Write a program that makes a square. (4 sides)

   LIST RUN

   _____

3. Write a program for a hexagon. (6 sides)

LIST RUN

---

4. Write a program for an octagon. (8 sides)

LIST RUN

---

5. Write a program for a nonagon. (9 sides)

LIST RUN

---

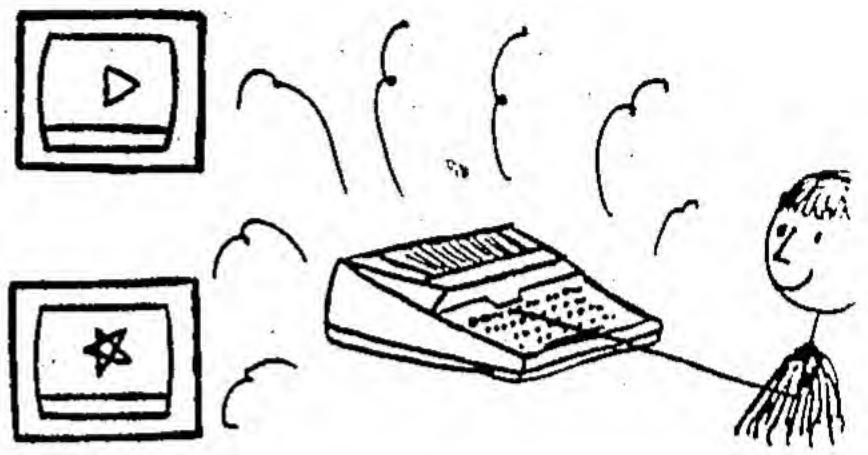
6. Write a program for a decagon. (10 sides)

LIST RUN

More MAGIC?

Turn a triangle

```
10 GR:3(DRAW 15;TURN 120)
```



into a star.

```
10 GR:5(DRAW 15;TURN 144)
```

Type this program.

```
10 GR:5(DRAW 15;TURN 144)
```

RUN

Now let's make more interesting stars.

A 7-pointed star

```
10 GR:7(DRAW 15;TURN 154)
```

Type this program.

```
10 GR:7(DRAW 15;TURN 154)
```

---

7. Make a 9-pointed star.

LIST RUN

_____

8. Make a 13-pointed star.

LIST RUN

_____

9. Make a 15-pointed star.

LIST RUN

_____

10. Make a 59-pointed star.

LIST RUN

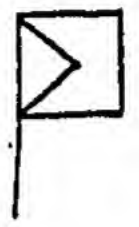
_____

11. Fill-in the missing parts of each program.

LIST

```
10 GR:3(DRAW 15;TURN ____)
20 __:4(____ 15;____ ____)
30 GR:TURN 180
40 __:____ 30
```

RUN



Now run the program.

12.

LIST

```
10 GR:5(____ __;____ ___)
20 __:TURN ___
30 ____________________
```

RUN

Run the program.

13. Make a program to draw a house.

LIST

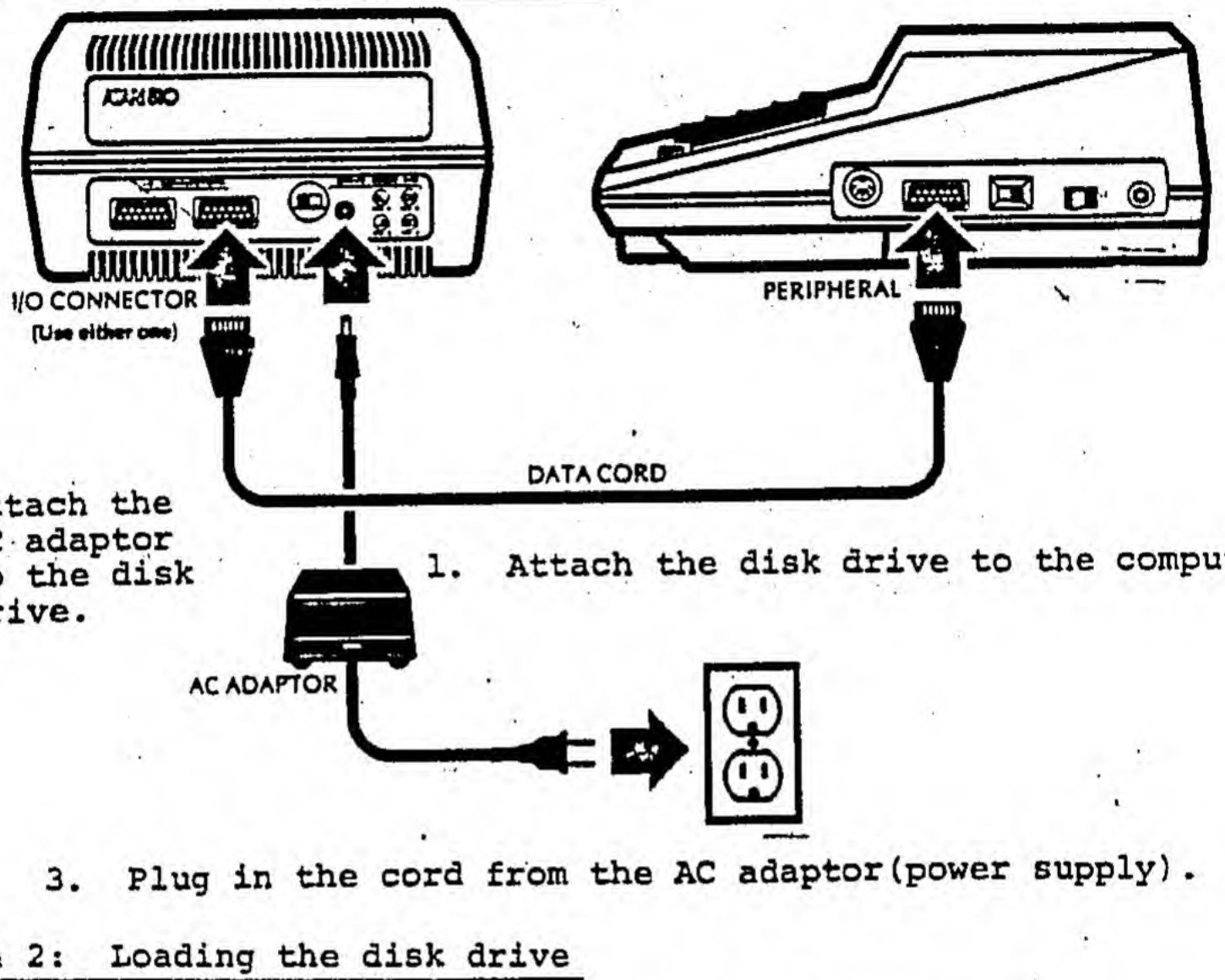
RUN

14. Now you may make any program that you would like. You can use anything that you know about in the PILOT language.

LIST RUN

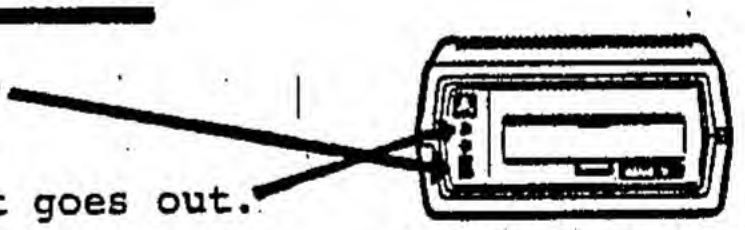
Use as many lines as you need. You can even use more lines. Use the next page as a workspace.

Section 1: Attaching the disk drive

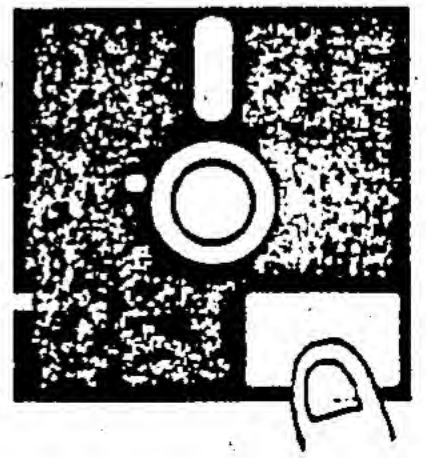


1. Attach the disk drive to the computer.
2. Attach the AC adaptor to the disk drive.
3. Plug in the cord from the AC adaptor(power supply).

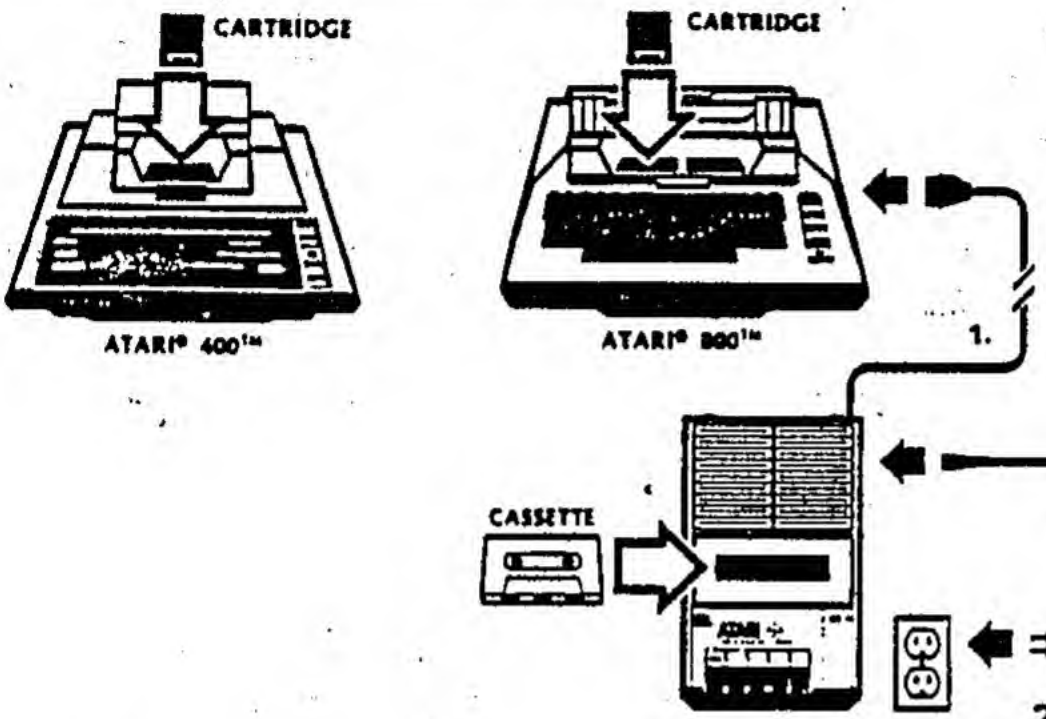
Section 2: Loading the disk drive


ATARI 810 DISK DRIVE

1. Turn on empty disk drive.
2. Wait until the busy light goes out.
3. Hold the disk on the label, with the label facing up.
4. Put disk in drive until you hear a click. Close door.
5. Turn on the television.
6. Turn on the computer.
7. Wait for the READY on the television screen.



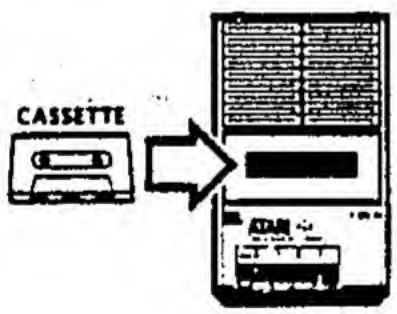
Section 1: Attaching the recorder



1. Attach the recorder to the computer.

2. Plug in the cord.

Section 2: Loading the cassette recorder



1. Insert the cassette.
2. Press REWIND.
3. When the tape stops, press STOP.
4. Set the counter to 000.

Programming in PILOT: Part 2

Section 1: Loading a program from a disk

1. Attach and load the disk drive.

2. Type LOADD: and the name of the program.

```
LOADD:TRISTAR (RETURN)

Do you see READY on the screen?
```

Section 2: Loading a program from a cassette

1. Attach and load the cassette recorder.

2. Find the beginning number of your program by pressing the ADVANCE key and the STOP key.

3. Turn on the television and the computer.

4. Type LOADC: . Press RETURN on the computer. It will beep.

5. Press the PLAY key on the recorder. Then press RETURN on the computer.

```
LOADC: (RETURN)

Press PLAY. Press (RETURN).

Do you see READY on the screen?
```

PA:

PA: means pause

GR:CLEAR

GR:PENRED

GR:PENBLUE

GR:PENYELLOW

Have you loaded TRISTAR or typed it into the computer?.

RUN

What did you see? a _____ then a _____

GR:QUIT
LIST

_____ 10 GR:3(DRAW 15;TURN 120)

_____ 20 PA:60

_____ 30 GR:CLEAR

_____ 40 GR:5(DRAW 15;TURN 144)

_____ 50 PA:60

_____ 60 GR:CLEAR

Think!

Where is the triangle made?

Put a triangle ▷ before that line number.

Put a star ☆ before the correct line number.

PA: means PAUSE

PA:60 is a one second pause.

Change line 20 to

```
20 PA:300
RUN
```

Change line 50 to

```
50 PA:5
RUN
```

Change the program so that you see the triangle for a short time and the star for a long time.

```
LIST
10 GR:3(DRAW 15;TURN 120)
20 PA: _____
30 GR:CLEAR
40 GR:5(DRAW 15;TURN 144)
50 PA: _____
60 GR:CLEAR
```

30

RUN

Think!

What happened when you took out line 30?

List the program.
Take out another line so that both the ▷ and the ★ stay on the screen.

What line? _____

RUN

Type lines 30 and 60 again so that your program looks like this.

```
10 GR:3(DRAW 15;TURN 120)
20 PA:60
30 GR:CLEAR
40 GR:5(DRAW 15;TURN 144)
50 PA:60
60 GR:CLEAR
```

Add a line so that the ▷ is red.

___ GR:PENRED

Add a line so that the ★ is blue.

___ GR:PENBLUE

Now, for more exciting programs.

```
40 GR:4(DRAW 15;TURN 90)

RUN
```

Think!

What happened to the program?

_____

Exercises

1. Change the ▷ to a ☆ .
   Change the □ to a ⬠ .

```
LIST

10 _____
20 PA:60
30 GR:CLEAR
40 _____
50 PA:60
60 GR:CLEAR
```

2. Change the program so that both shapes are larger.

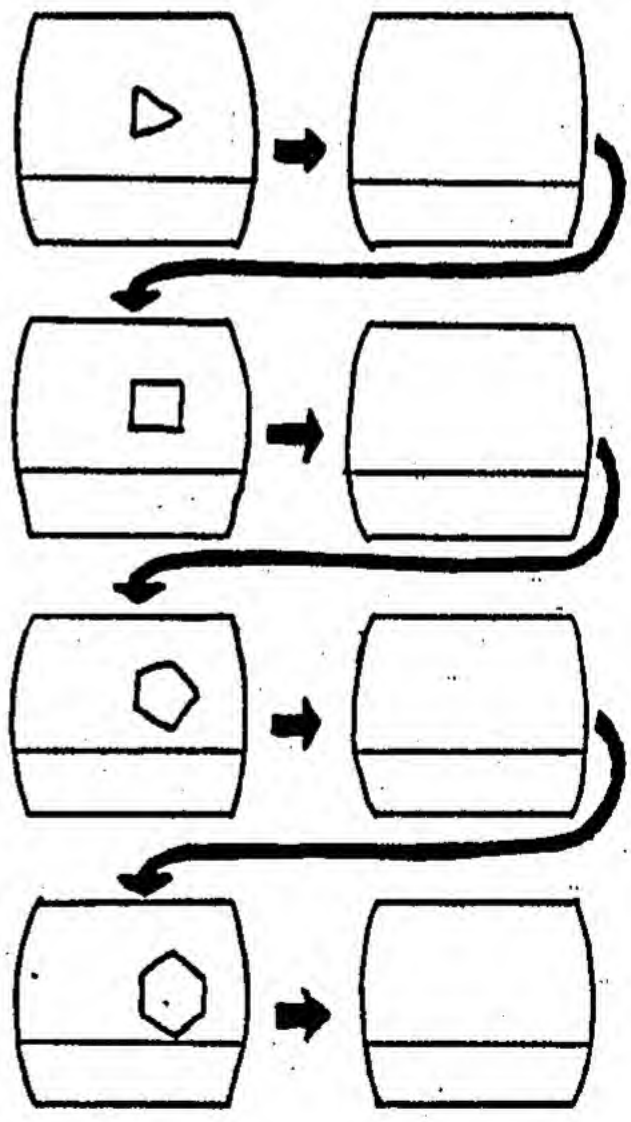
```
LIST

10 _____
20 PA:60
30 GR:CLEAR
40 _____
50 PA:60
60 GR:CLEAR
```

3. Make both shapes stay on the screen.

```
LIST
_____________________________
_____________________________
_____________________________
_____________________________
```

4. Fill in the missing parts of this program.
It draws a ▷ , looks at it, then clears it away.
Then it draws a □, looks at it, then clears it away.
Then it draws a ⬠, looks at it, then clears it away.
Then it draws a ⬡, looks at it, then clears it away.

```
LIST
10 GR:3(_____________________________)
20 PA:60
30 GR:CLEAR
40 GR:4(_____________________________)
50 _____________________________
60 _____________________________
70 _____________________________
80 _____________________________
90 _____________________________
100 GR:6(_____________________________)
110 _____________________________
120 _____________________________
```



Run the program.

Write a program of your own.

Use any of these commands or any other command you know.

GR:TURN __ PA: __ GR:PENRED GR:PENYELLOW
GR:DRAW __ GR:CLEAR GR:PENBLUE

LIST RUN

* names a label

J: jumps to a label

NEW

Load TRISTAR into the computer from a disk or a cassette.

LIST

Add these lines.

```
5 *LOOP
70 J:*LOOP

RUN
```

Think!

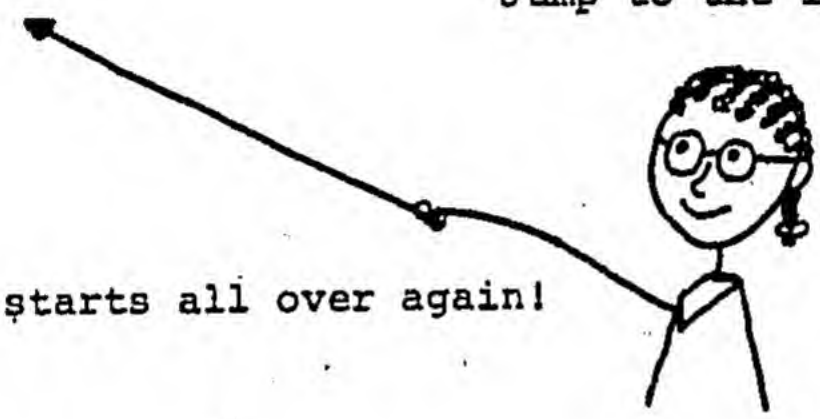
What's happening?

It won't stop. Press BREAK.

Run the program again.
Stop the program again.

Let's read the program like the computer does.

```
5 *LOOP                          A label for the part that follows
10 GR:3(DRAW 15;TURN 120)        Draw a triangle
20 PA:60                         Wait one second
30 GR:CLEAR                      Clear the screen
40 GR:5(DRAW 15;TURN 144)        Draw a star
50 PA:60                         Wait one second
60 GR:CLEAR                      Clear the screen
70 J:*LOOP                       Jump to the label
```



And then it starts all over again!

And again!

And again! . . .

Exercises

1. Change the program to ☆ and ⬠.

LIST

5 _____

10 _____

20 _____

30 _____

40 _____

50 _____

60 _____

70 _____

2. Make your own change in the program.

LIST

5 _____

10 _____

20 _____

30 _____

40 _____

50 _____

60 _____

70 _____

Do you want to save your program?
Look at the appendix on Saving Programs to see how.

3. Make a program with a flashing star.
(Hint: Make a star. Pause a little. Then make the star disappear. Do this again and again.)

_____

_____

_____

_____

_____

4. Create a program of your own. You may have as many lines as you want in your program.

LIST

Try this if you like.

Ready to be fancy? Let's make a design.

Here is a program.

```
*DESIGN ————————————————————→ *DESIGN
Draw a triangle ————————————→ GR:3(DRAW 15;TURN 120)
Turn 10 to the right ———————→ GR:TURN 10
Jump to the beginning ——————→ J:*DESIGN
```

Type your program in .

```
10 *DESIGN
20 GR:3(DRAW 15;TURN 120)
30 GR:TURN 10
40 J:*DESIGN
```

or

```
AUTO
0
*DESIGN
GR:3(DRAW 15;TURN 120)
GR:TURN 10
J:*DESIGN
```

What other designs can you make?

LIST

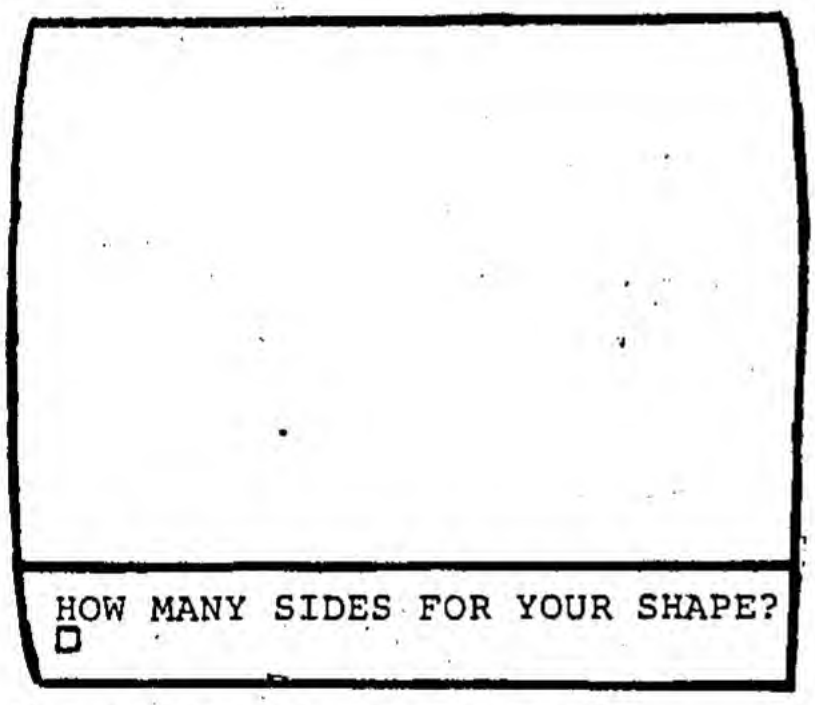
Section 1: A new command and a new shape

GR:GO

Load SHAPELY into the computer.

RUN

Do you see



on the bottom of the screen?

Type in a 3. Press (RETURN) .

What do you see on the graphics screen? _____

Type in these numbers.

Fill in the chart.

	NUMBER OF SIDES	TURN NUMBER	SHAPE
3	3	120	▷
4	4		
5			
6			
8			
9			
10			
12			

What is happening on the screen?

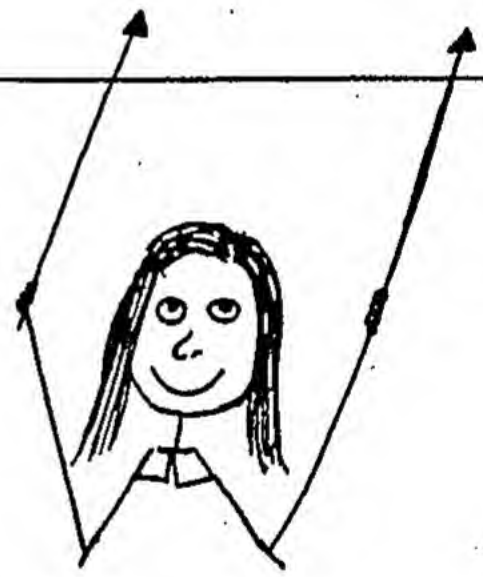
_____

_____

What is happening on the chart?

_____

_____



Type in 18.

Press (BREAK)

NEW

```
GR:CLEAR

10 GR:PENRED;18(DRAW 10;TURN 20)

RUN
```

This is an 18 sided shape.

It looks a lot like a circle.

Make a 36 sided shape.

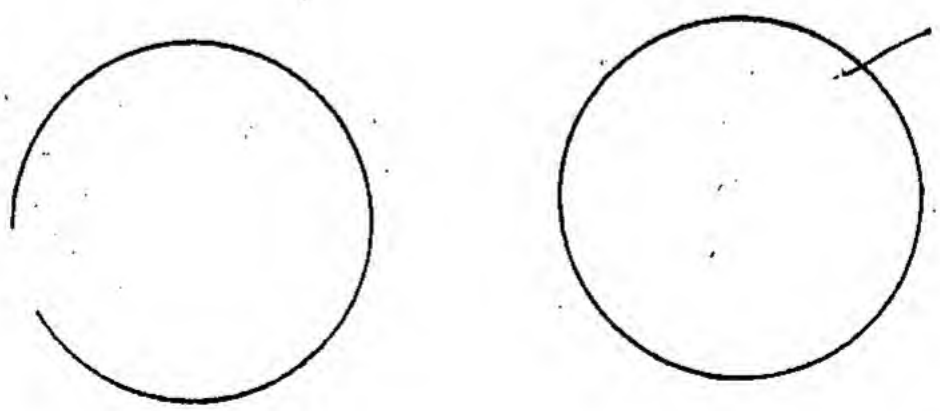
```
LIST

20 GR:PENYELLOW;36(DRAW 10;TURN _____)
```

Find the <u>smallest</u> possible turn number.

Think!

Should this number be more or less than the 18 sided shape?

The number is correct when the turtle comes back to the starting point <u>and</u> stops.

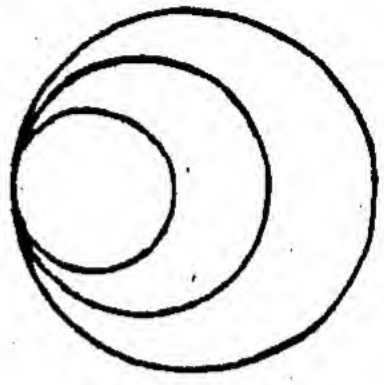


Exercises

1. Write a program to make circles touching at a point.

LIST

RUN



2. Make each circle a different color.
   Put only 3 lines in your program.

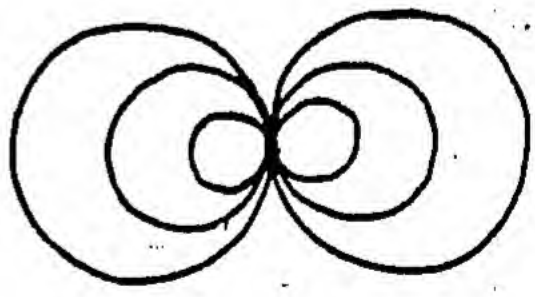
LIST

RUN

3. Make owl eyes.

LIST

RUN



```
NEW

GR:CLEAR

GR:GO 5

GR:GO 5
```

Think!

What is happening?

```
GR:GO 10

GR:GO 5;GO 5

GR:2GO 5

GR:PENBLUE;GO -10

GR:PENRED;2GO -7

10 GR:4(DRAW 10;TURN 90)

RUN

LIST

20 GR:GO 20;4(DRAW 10;TURN 90)
```

Change line 20 to

```
20 GR:GO 20;4(GO 10;TURN 90)
```

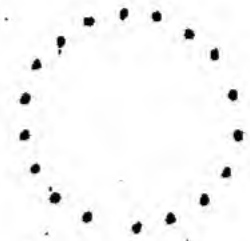
Exercises

1. Make a circle that looks like this.

LIST

_____

RUN



2. Make a square that looks like this.

LIST

_____

RUN

3. Fill in the missing parts of the program.

LIST

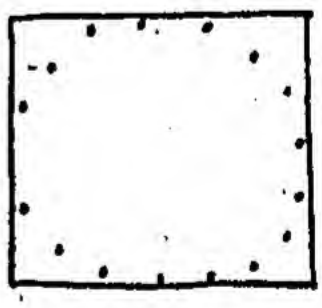
____ GR:GO _____

____ GR:4(_____ 60;_____ ____)

____ GR:. _____ ____

____ GR: 18(_____ ____; _____ _____)

RUN



4. Create a program of your own using the GO command.

LIST RUN

T: R: C:

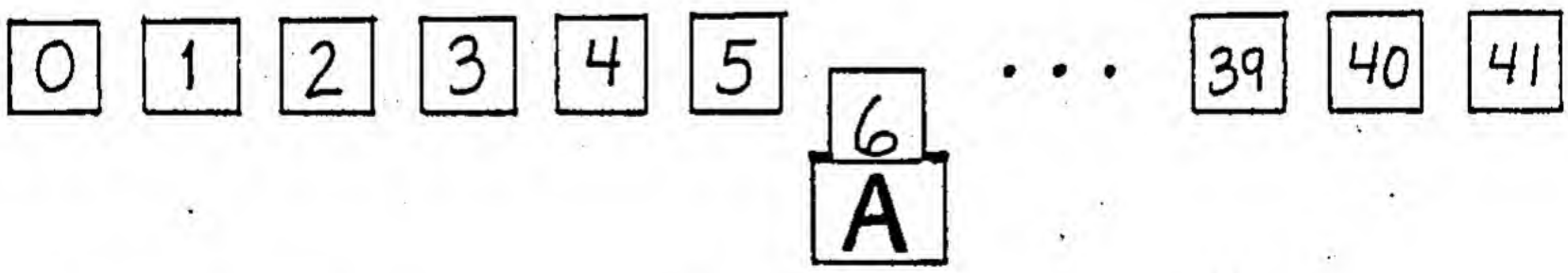
T: means type | R: means remark | C: means compute

```
10 GR:PENRED
20 GR:18(GO 4;TURN 20)

RUN
```

```
5 *AGAIN
30 GR:TURN 5
40 J:*AGAIN

RUN
```

BREAK

We will put a counter in the program.
We will only make the circle 41 times.



List the program.

Add these lines

```
4 C:#A=0
35 C:#A=#A+1
```

Change line 40

```
40 J(#A<41):*AGAIN
```

```
RUN
```

List the program.

Let's read the program like the computer does.

```
4 C:#A=0                          Lets A be 0 in the computer's memory
5 *AGAIN                          A label for what follows
10 GR:PENRED                      Turtle picks up a red pen
20 GR:18(GO 4;TURN 20)            Draws a dotted circle
30 GR:TURN 5                      Turns a little
35 C:#A=#A+1                      Lets A be the old A plus 1
40 J(#A<41):*AGAIN                Jumps to the label as long as A is
                                        less than 41
```

```
RUN
```

Was the run different? _____

R:CIRCLE TO HEART is just a programmer's remark. It is like a note to remind the programmer of something that is important to remember.

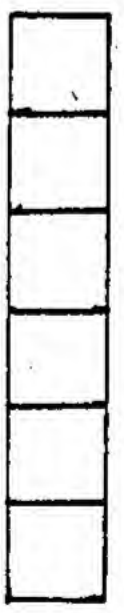
```
50 T:                    I LOVE YOU
RUN
```

Exercises

1. Fill in the missing parts of this program to make an apartment building.

LIST RUN

```
____ ____:BUILDING
____ GR: GO _______
____ C:#S= _____
____ *STORY
____ GR: ____________________________
____ GR: _____ _______
____ C:#S= ___________________
____ J(_____):_____________________
```



2. Write a program to make a design. Add a counter to the program so it stops when the design is done.

LIST RUN

_____

_____

_____

_____

_____

_____

_____

_____

3. Fill in the missing parts of the program to make a car.

LIST RUN

____ R:_____

____ GR:2(_____)

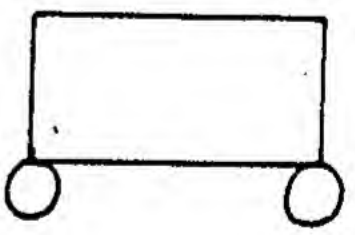
____ R:_____

____ GR:TURN_____

____ GR:18(DRAW _____;TURN _____)

____ GR:GO _____

____ GR:18(DRAW _____;TURN _____)



4. Write a program to type your name 100 times.

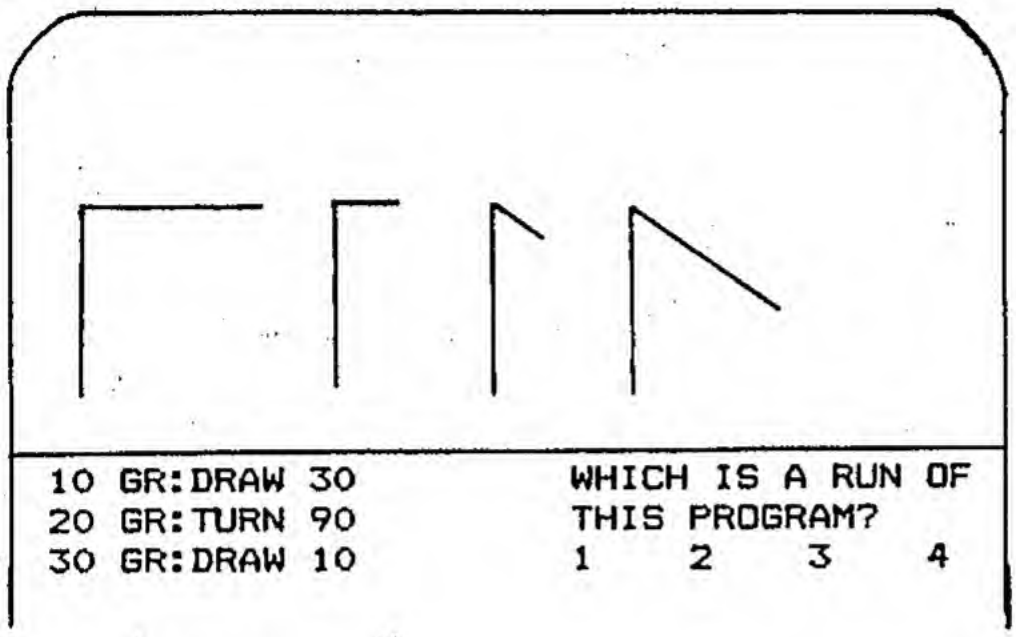
LIST RUN

5. Write a program using remark statements (R:) and a loop with a counter.

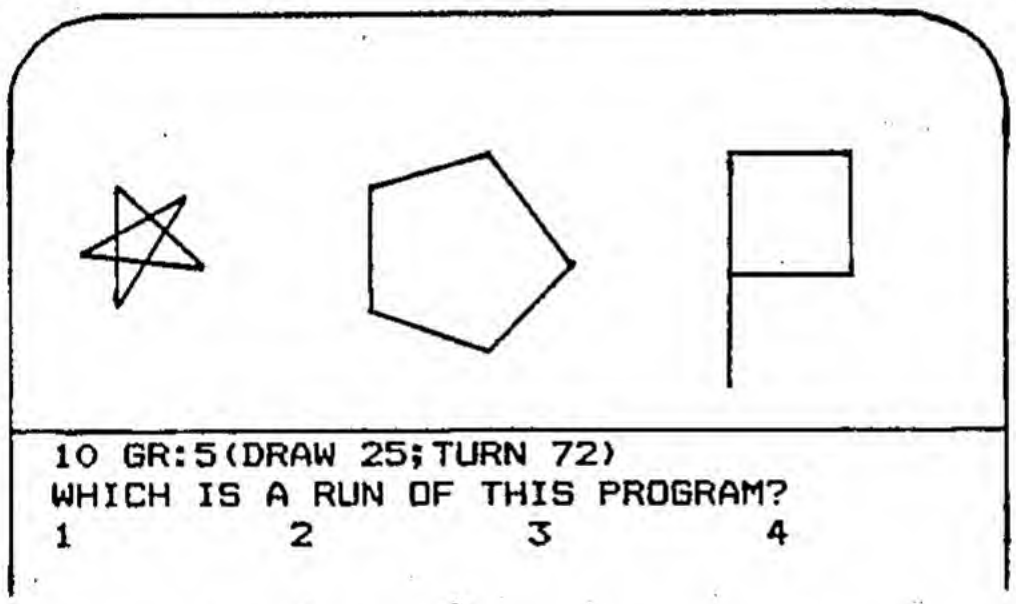
LIST RUN

Appendix B: Screen Displays/Multiple-choice Computer Tests

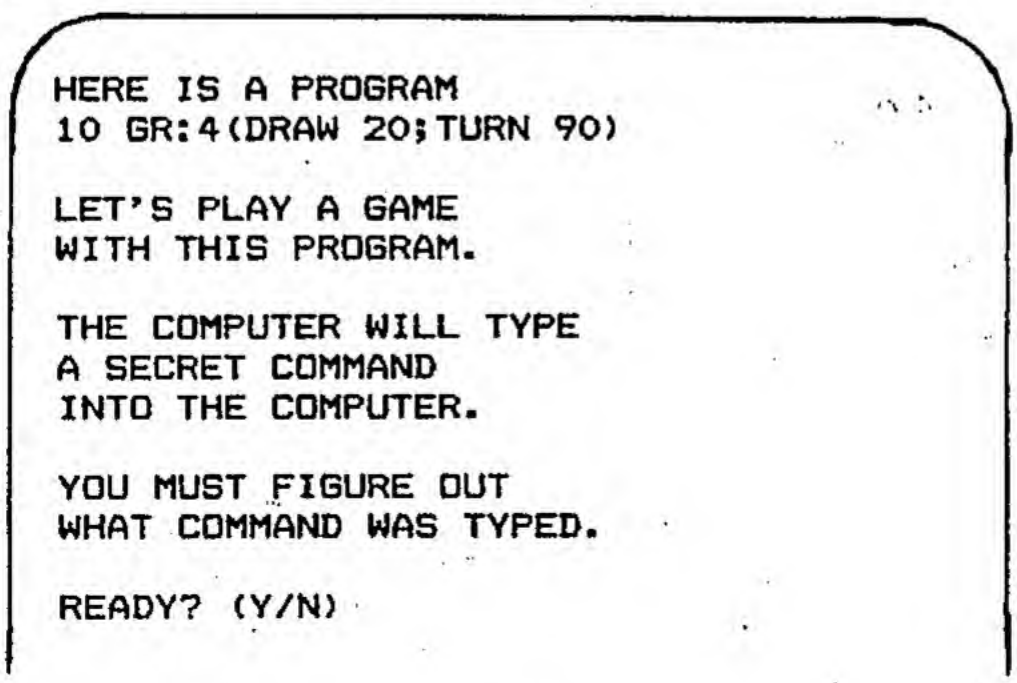
Test 1 - Item 1



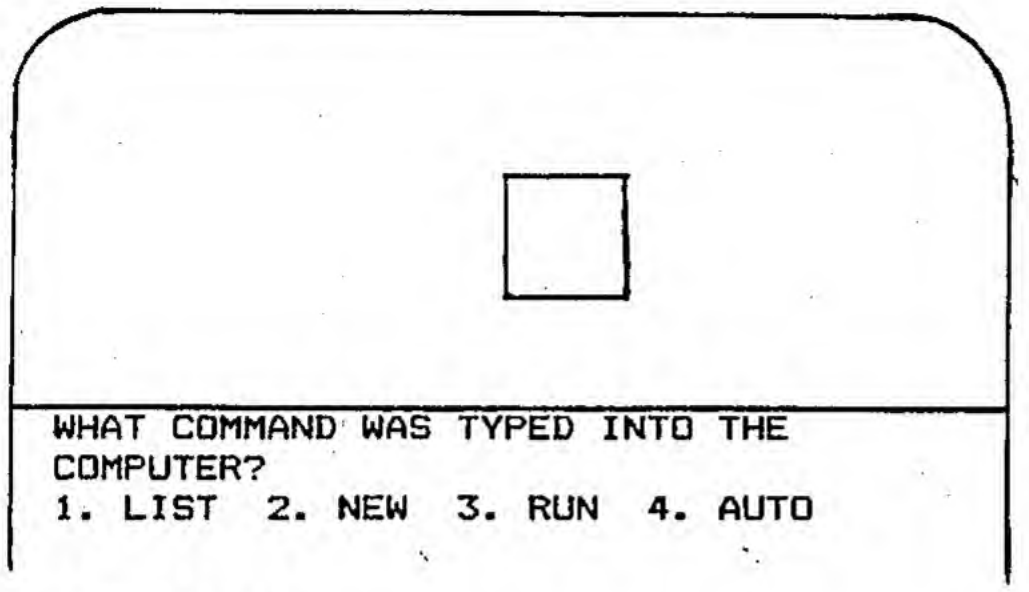
Test 1 - Item 2



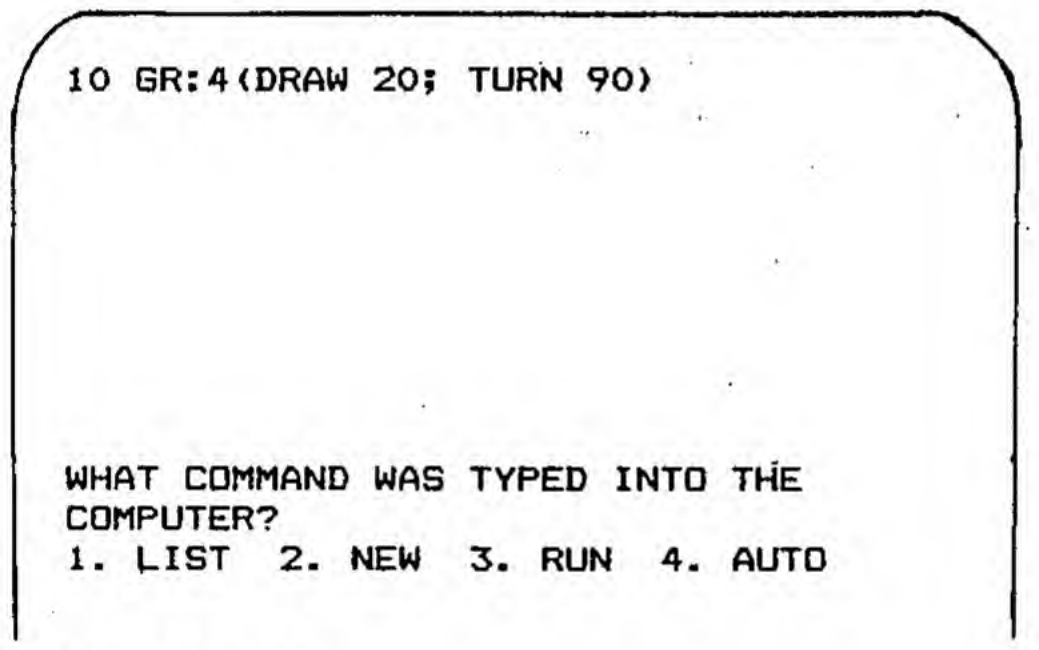
Test 1 - Screen display before Item 3



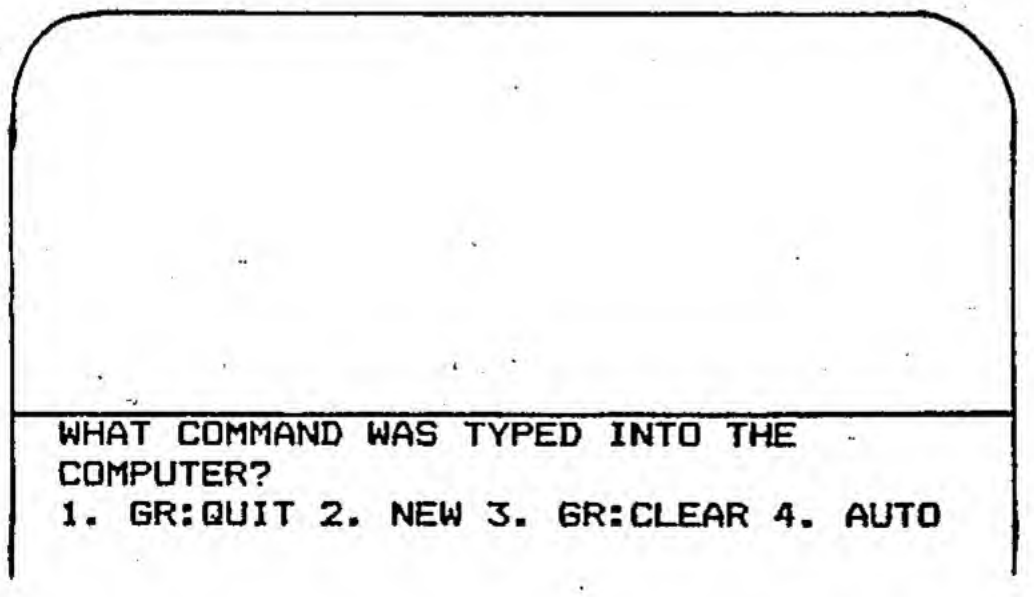
Test 1 - Item 3



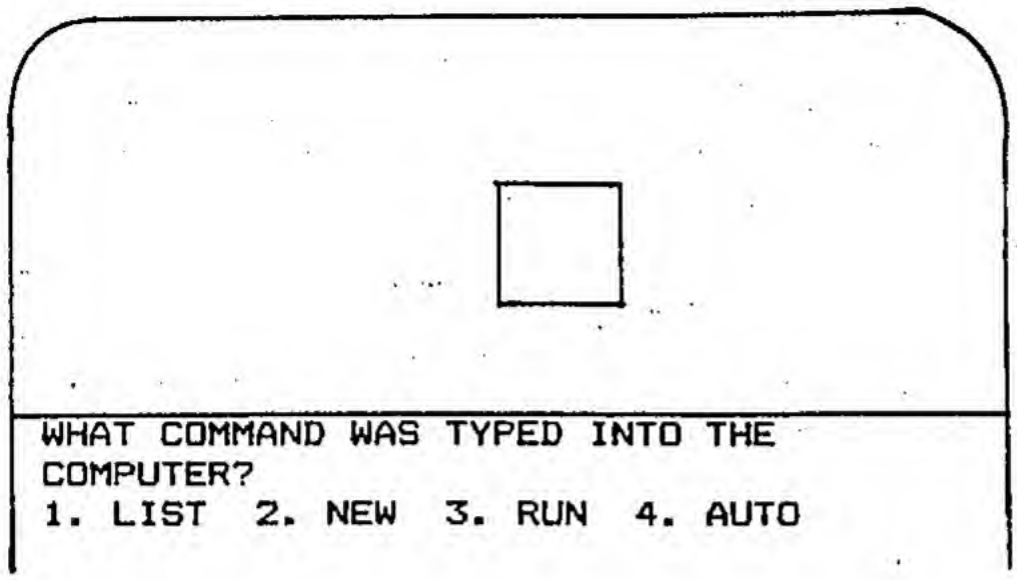
Test 1 - Item 4



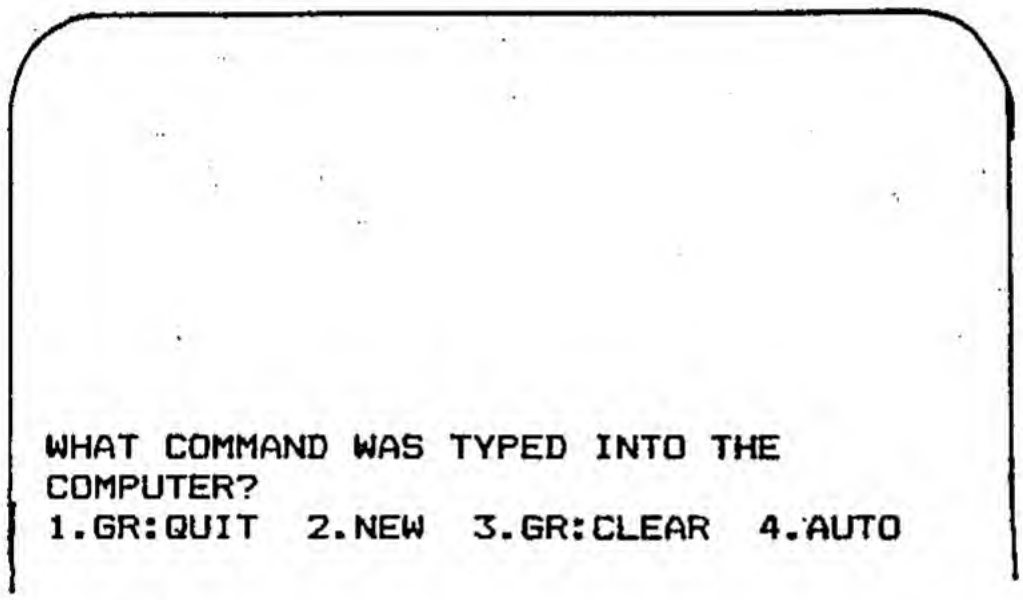
Test 1 - Item 5



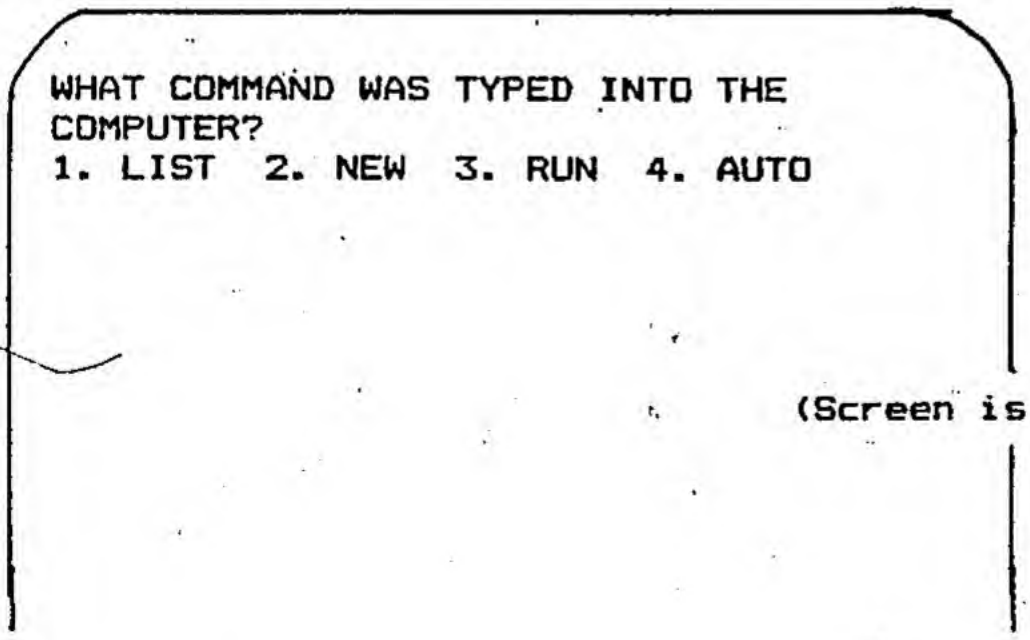
Test 1 - Item 6



Test 1 - Item 7



Test 1 - Item 8



Test 2 - Item 1

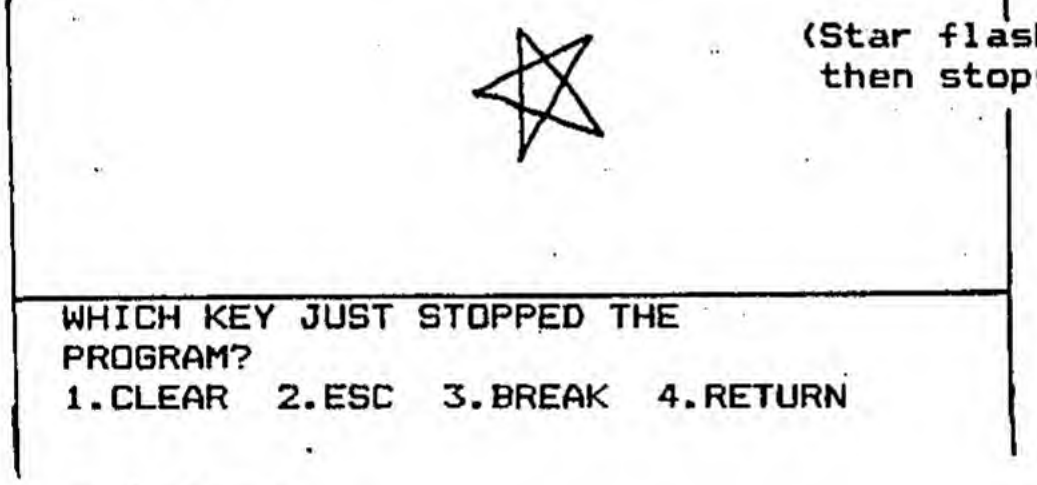
HERE IS A PROGRAM.

```
5 *LOOP
10 GR:5(DRAW 25;TURN 144)
20 PA:60
30 GR:CLEAR
40 J:*LOOP
```

WHEN YOU KNOW WHAT THE PROGRAM
WILL DO, RUN THE PROGRAM.

(Needs to type in RUN)

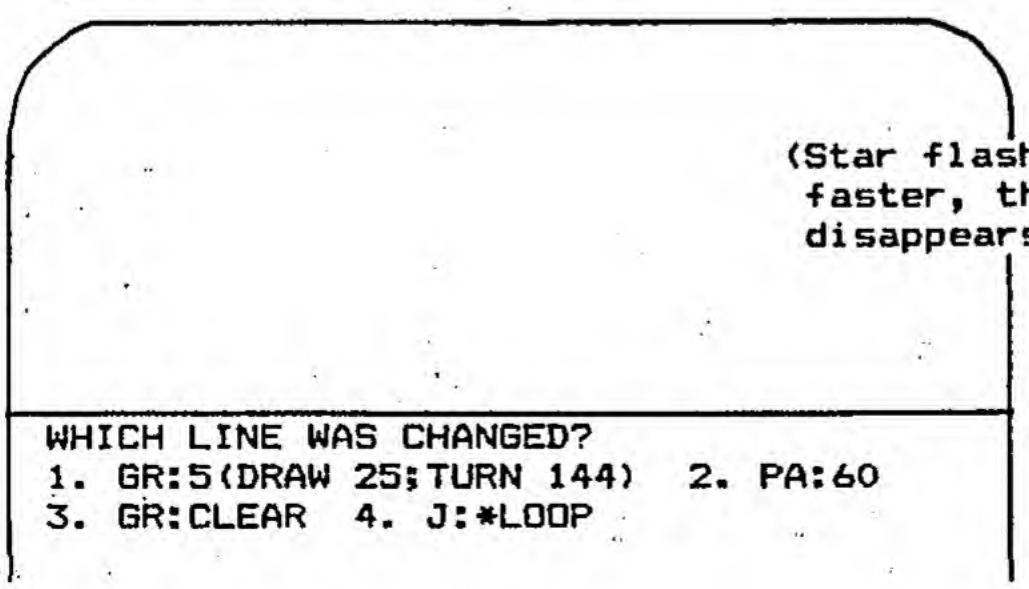
Test 2 - Item 2



WHICH KEY JUST STOPPED THE
PROGRAM?
1.CLEAR 2.ESC 3.BREAK 4.RETURN

Test 2 - Item 3

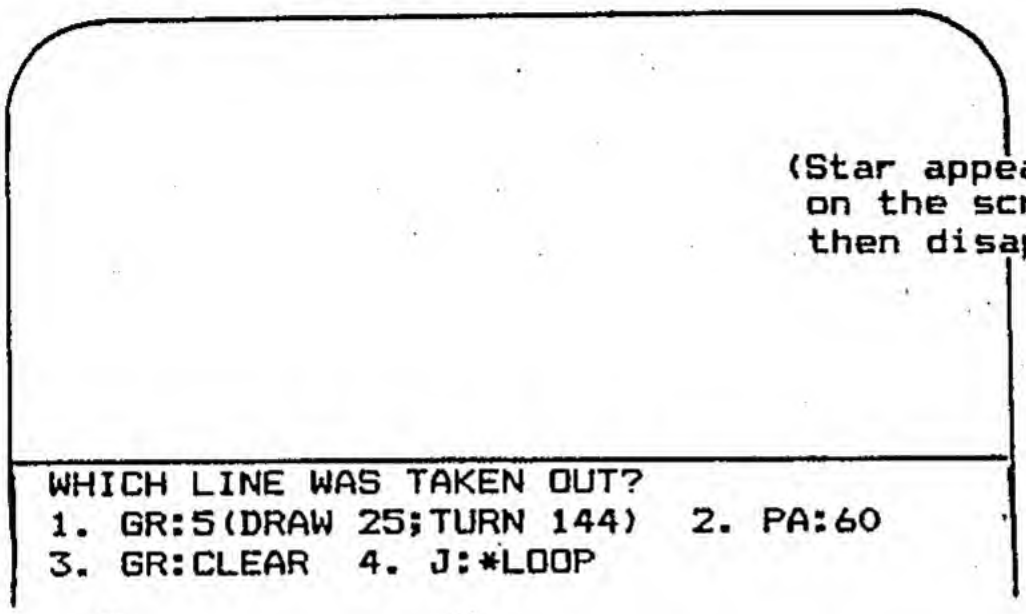
THERE IS A CHANGE IN THE PROGRAM.
WATCH CAREFULLY!



WHICH LINE WAS CHANGED?
1. GR:5(DRAW 25;TURN 144) 2. PA:60
3. GR:CLEAR 4. J:*LOOP

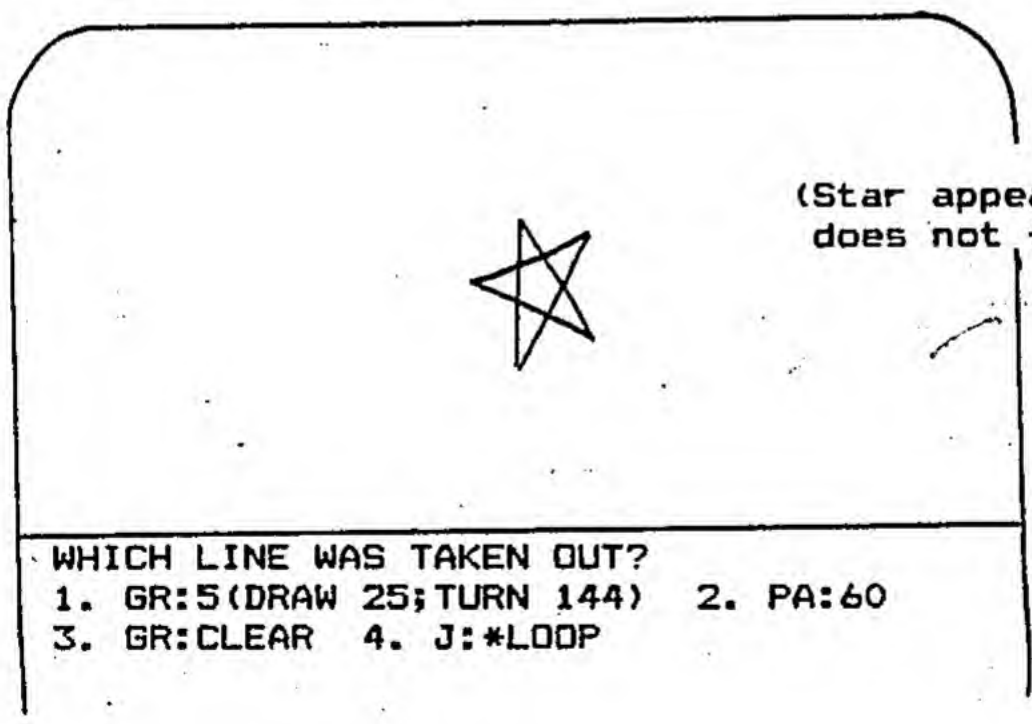
Test 2 - Item 4

A LINE HAS BEEN TAKEN OUT OF
THE PROGRAM.
WATCH CAREFULLY!

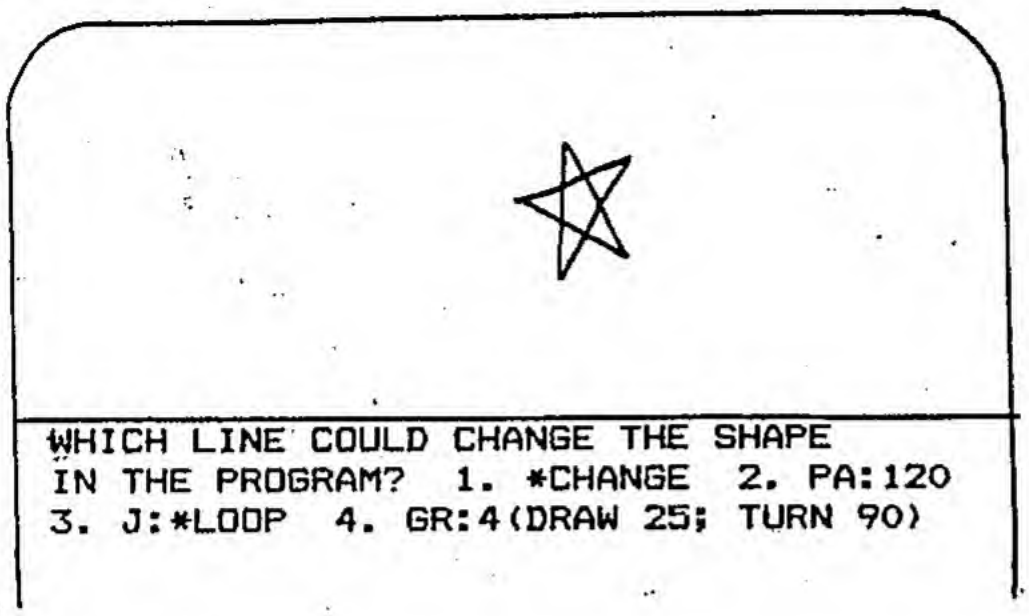


Test 2 - Item 5

A DIFFERENT LINE HAS BEEN TAKEN OUT.
WATCH CAREFULLY!



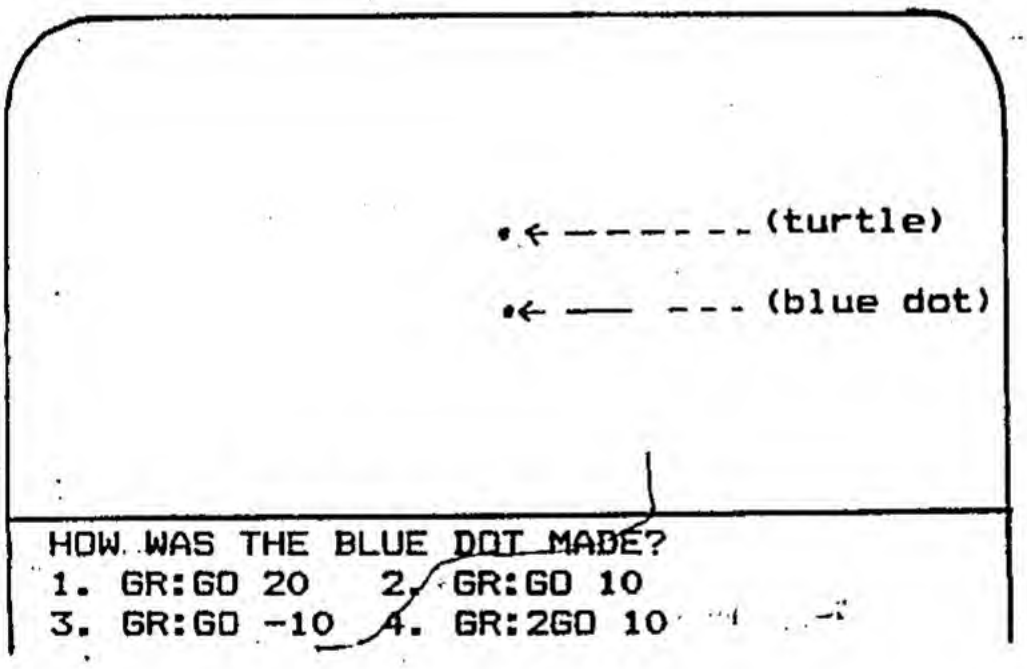
Test 2 - Item 6



WHICH LINE COULD CHANGE THE SHAPE
IN THE PROGRAM? 1. *CHANGE 2. PA:120
3. J:*LOOP 4. GR:4(DRAW 25; TURN 90)

Test 3 - Item 1

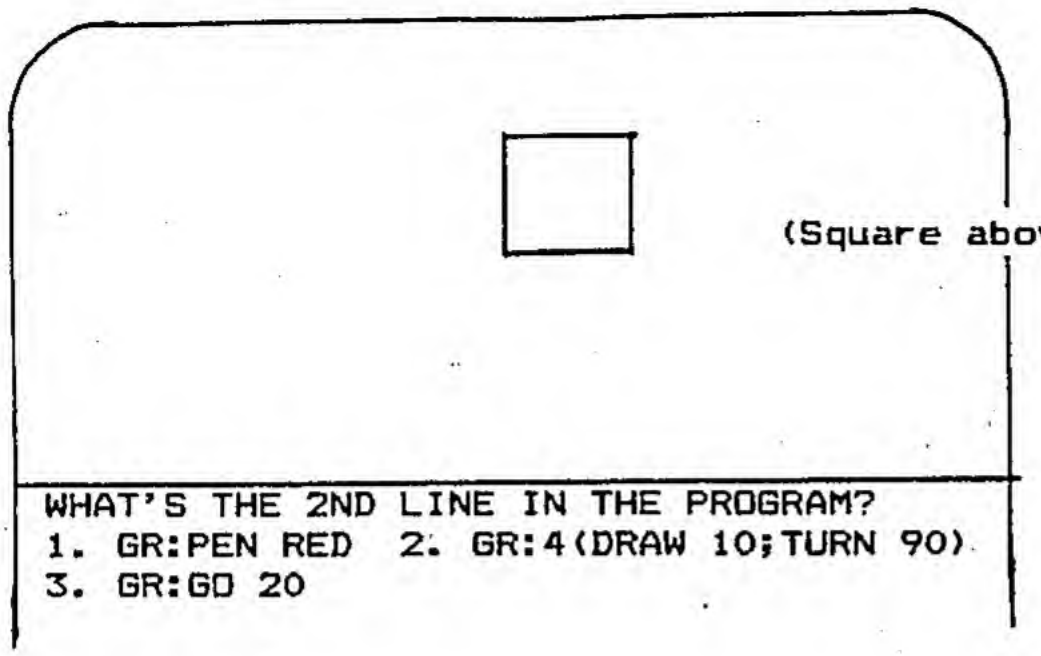
HERE IS THE TURTLE. (pause)



HOW WAS THE BLUE DOT MADE?
1. GR:GO 20 2. GR:GO 10
3. GR:GO -10 4. GR:2GO 10

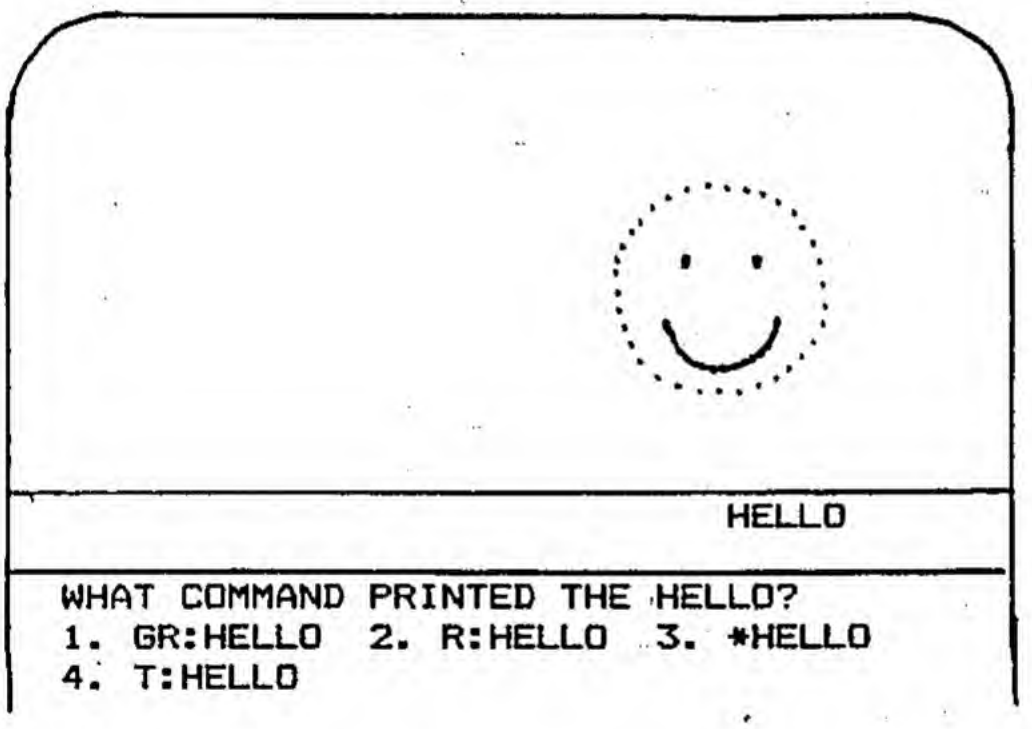
Test 3 - Item 2

WATCH CAREFULLY!
YOU WILL SEE THE RUN OF A PROGRAM.
WHEN YOU ARE READY, PRESS RETURN.



Test 3 - Item 3

WATCH THIS PROGRAM!



Test 3 - Item 4

WATCH CAREFULLY!

```
WHOOPEE  WHOOPEE  WHOOPEE  WHOOPEE
WHOOPEE  WHOOPEE  WHOOPEE  WHOOPEE
WHOOPEE  WHOOPEE  WHOOPEE  WHOOPEE
WHOOPEE  WHOOPEE  WHOOPEE  WHOOPEE
WHOOPEE  WHOOPEE  WHOOPEE  WHOOPEE
WHOOPEE  WHOOPEE  WHOOPEE  WHOOPEE
WHOOPEE  WHOOPEE  WHOOPEE  WHOOPEE
WHOOPEE  WHOOPEE  WHOOPEE  WHOOPEE
WHOOPEE  WHOOPEE  WHOOPEE  WHOOPEE
```

WHAT IS THE 3RD LINE IN THE PROGRAM?
1. J(#P<100):*LOOP 2. *LOOP
3. T:WHOOPEE \ 4. C:#P=#P+1

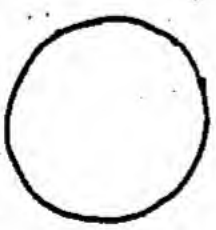
Test 3 - Item 5

Look at the screen!

(Triangle made, then square, etc.)

WHAT IS GETTING LARGER?
1. DRAW NUMBER 2. TURN NUMBER
3. NUMBER OF SIDES

Test 3 - Item 6



(Circle made to the left of center.)

HOW WAS THIS CIRCLE MADE?
1. 18(DRAW 10;TURN 20) 2. 18 (DRAW 5;TURN -20)
3. 18(DRAW 5;TURN 20) 5. 18 (DRAW 10;TURN -20)

Appendix C: Copy of Individual Student Achievement Graph

Individual Student Achievement Graph

Name _____ Teacher _____

Grade _____

Date | Section completed

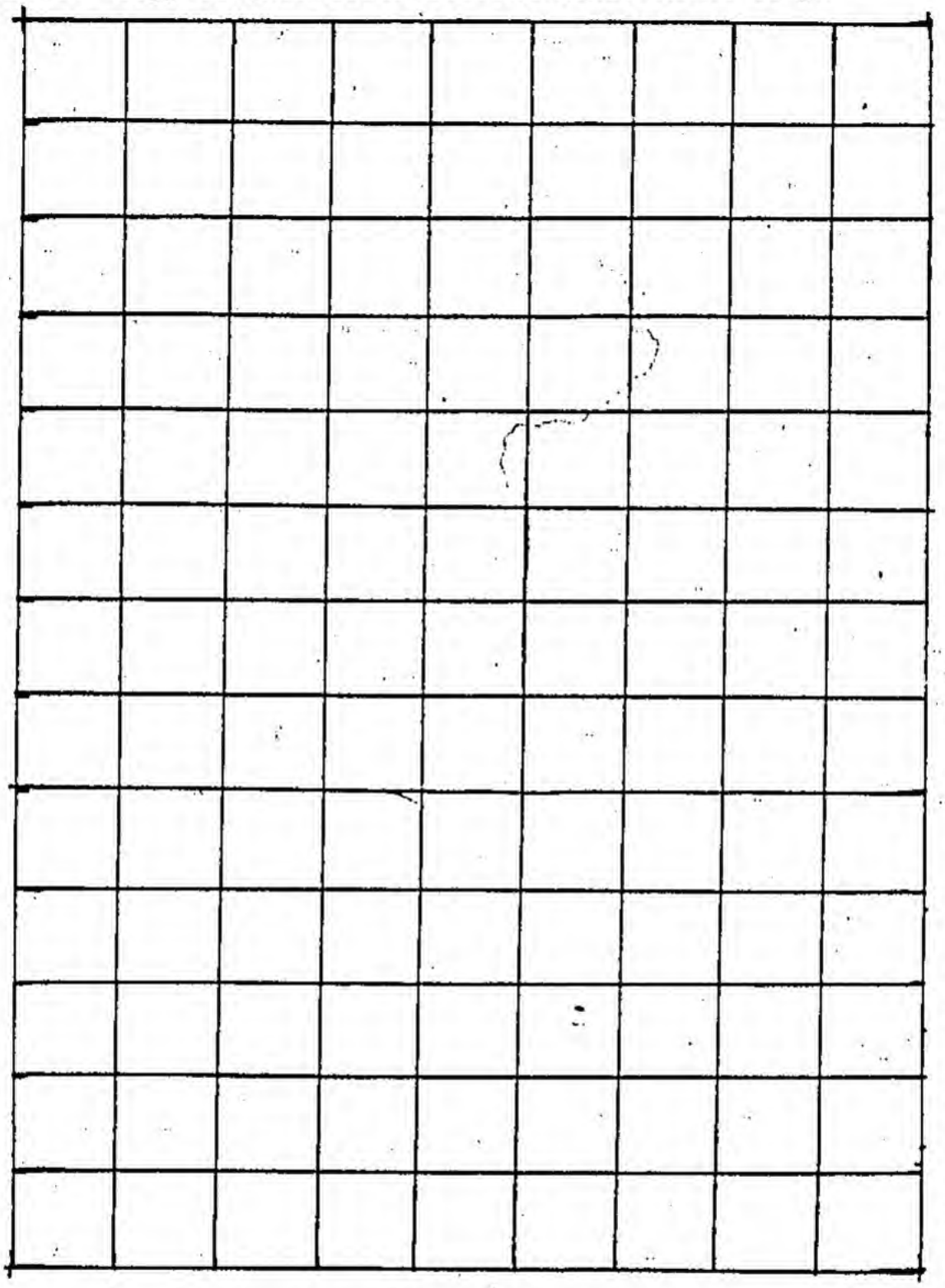
Test Scores

Part 1 _____

Part 2 _____

Part 3 _____

Part 4 _____



Appendix D: Teacher Form

Week of .....

Teacher's name .....

Number of students .....

Teacher Assistance

When the student asks for help, put a tally mark in the appropriate b
to indicate the kind of help he/she needed.

Reading/ Reading Underst'ng	Machine Operation	Programming/ Computing needs	Help with finding or correcting errors	Mathematical or number/ related	Other

Student Assistance

Do students assist each other? .....
How often (this week)? (50 % of the time, 75 %, etc.) .....

Work Time

Independent work time this week (each student) .....

Significant Problems

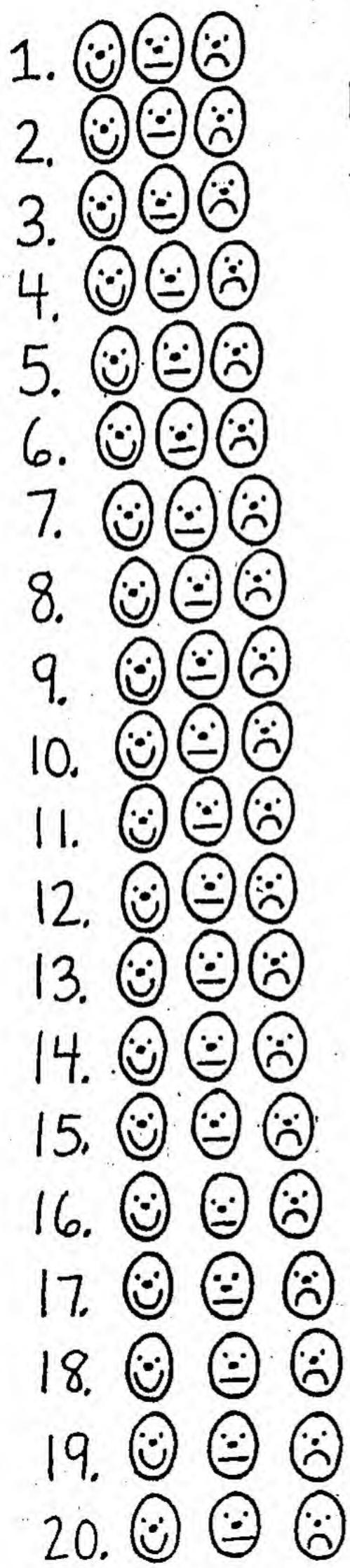
Teacher

Student

Other Comments

Appendix E: Student Response Sheet for Attitude Survey

Name _____



Name _____ Grade _____

A. Learning materials

Part	Date completed	Score
1.1		
1.2		
1.3		
2.3		
2.4		
3.1		
3.2		

Last section completed by 2/11 _____

B. Multiple-choice computer tests

Part	Date taken	Score
Part 1		
Part 2		
Part 3		

C. Programming tests

Part	Date taken	Success
Part 1		
Part 2		
Part 3		

D. Rating of programming ability _____

E. Attitude survey

Attitude	Raw score	Percentile
General		
Writing		
Editing		
Aid		
Self-confidence		
Motivation		

Appendix G: Final Teacher Evaluation Form

1. As completely as possible, describe the computer program in your classroom these past eight weeks.

2. Describe your own feelings on having the computer in your classroom.

3. Rate the present effectiveness of the learning materials in the following areas. (Mark 1 to 5; 1 being very effective.)

	1	2	3	4	5
Clarity (to student)	I	I	I	I	I
Clarity (to teacher)	I	I	I	I	I
Ease of use	I	I	I	I	I
Reading ease	I	I	I	I	I
Independence	I	I	I	I	I

Appendix H: Questionnaire for Parents on Computers in the Home

Your name.....

Your child's name.....

If you do not have a computer in your home, you need only answer questions 1, 10, 11, and 12.

1. Do you have a computer in your home?.....

2. What kind of computer?.....

3. What functions does it serve? (word processing, education, recreation, other).....

4. Does your son/daughter use the computer?.....

5. Alone? with others?.....

6. For what purpose(s)?.....

7. Does he/she help you (a parent) and/or any other member(s) of his/her family?.....

8. Can your child program the computer?.....

9. In what programming language?.....

10. Are you considering the purchase of a computer in the near future?.....

11. Does anyone in your immediate family work in any way with computers?.....

12. If yes, who, and in what capacity?.....
.....

Thank you for completing this questionnaire. Its completion is of utmost importance in a research study I will be conducting this year.

Nancy A. Shuller
Primary Computer Specialist
The Day School